Managing Aegis System Software

010852-A00

apollo

Managing Aegis System Software

Order No. 010852-A00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

This manual is dedicated to the memory of Kriss Kellerman,
a valued friend and colleague.

Preface

Managing Aegis System Software describes system administration in the Aegis™ environment. We've organized this manual as follows:

Chapter 1	Introduces system administration in the Aegis environment and provides a description of changes in Software Release 10 (SR10).
Chapter 2	Describes how to maintain nodes and provide services, including procedures to catalog nodes and manage root directories with **ns_helper**.
Chapter 3	Provides a comprehensive discussion of the network environment, including both start-up procedures and files and server reference information.
Chapter 4	Describes registries, the **/etc/passwd** and **/etc/group** files, and how to update and replicate registry information.
Chapter 5	Discusses system and software security and the protection system.
Chapter 6	Provides descriptions of Aegis administration commands. This chapter also contains manual pages for selected TCP/IP commands. Refer to *Configuring and Managing TCP/IP* for descriptions of all TCP/IP commands.
Appendix A	Discusses how to use **netmain** and **netmain_svr**.

Intended Audience

This manual is intended for users familiar with Aegis software, the Domain®/OS system, and the UNIX* operating system.

The best introduction to the Domain/OS system is *Getting Started With Domain/OS* (Order No. 002348). This manual explains how to use the keyboard as well as display, read, and edit text, and manipulate files. It also shows how to request Domain/OS services by using interactive commands.

*UNIX is a registered trademark in the USA and other countries.

If you are not familiar with the UNIX operating system, we recommend that you read one of the following documents:

- Bourne, Stephen W. *The UNIX System.* Reading: Addison-Wesley, 1982.
- Kernighan, Brian W. and Rob Pike. *The UNIX Programming Environment,* Englewood Cliffs, Prentice-Hall, 1984.
- Thomas, Rebecca and Jean Yates. *A User Guide to the UNIX System.* Berkeley: Osborne/McGraw-Hill, 1982.

Related Manuals

The Help file **manuals** lists current revisions of all manuals for this software release.

Refer to the *Domain Documentation Quick Reference* (Order No. 002685) for a complete list of related documents. For more information on managing the Aegis environment, refer to the following documents:

- *Using Your Aegis Environment* (Order No. 0011021).
  This is the first volume you should read. It explains how Aegis works, and describes the Bourne shell, C shell, and Mail.
- *Making the Transition to SR10 Operating System Releases* (Order No. 011435).
  This book describes how to make the transition from Software Release 9.7 (SR9.7) of the Domain operating system to Software Release 10 (SR10). It includes an overview of new features and discusses the operation of a network of mixed-release (SR9.x and SR10) machines.
- *Managing Domain Routing and Domain/OS in an Internet* (Order No. 005694).
  This book describes managing Domain/OS software in an internet environment.
- *Installing Software with Apollo's Release and Installation Tools* (Order No. 008860).
  This book describes how to install standard software and optional products, as well as how to configure software and initialize a disk.
- *Aegis Command Reference* (Order No. 002547).
  This book describes all the shell commands supported by Aegis.
- The *Domain/OS Call Reference, Volumes 1 and 2* (Order Nos. 007196 and 012888).
  This book describes calls to operating system components that are accessible to user programs.
- *Programming with Domain/OS System Calls* (Order No. 005506).
  This book describes all the Aegis system calls and library functions.
- The *Domain C Language Reference* (Order No. 002093).
  This book describes C program development on the Domain system. It lists the features of C, describes the C library, and gives information about compiling, binding, and executing C programs.
- *Domain Display Manager Command Reference* (Order No. 011418).
  This book describes the Display Manager commands used by Domain/OS in all three environments (Aegis, SysV, and BSD).

- *Using TCP/IP Network Applications* (Order No. 008667).
  This book provides instructions for using commands on the TCP/IP network.
- *Configuring and Managing TPC/IP* (Order No. 008543).
  This book explains how to configure and manage a TCP/IP internet.
- *DPSS/Mail User's Guide* (Order No. 003660).
  This book provides instructions for starting and using DPSS/Mail™.
- Printing in the Aegis Environment (Order No. 011774).
  This book provides instructions for printing documents in the Aegis environment.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. To make it easy for you to communicate with us, we provide the Apollo Problem Reporting (APR) system for comments related to hardware, software, and documentation. By using this formal channel, you make it easy for us to respond to your comments.

You can get more information about how to submit an APR by consulting the appropriate Command Reference manual for your environment (Aegis, BSD, or SysV). Refer to the **mkapr** (make apollo problem report) shell command description. You can view the same description online by typing:

$ **man mkapr** (in the SysV environment)

% **man mkapr** (in the BSD environment)

$ **help mkapr** (in the Aegis environment)

Alternatively, you may use the Reader's Response Form at the back of this manual to submit comments about the manual.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term.
user-supplied values	*Italic words or characters in formats and command descriptions represent values that you must supply.*
example user input	In examples, information that the user enters appears in color.
`output`	`Information that the system displays appears in this typeface.`

[ ]	Square brackets enclose optional items in formats and command descriptions.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
CTRL/	CTRL/ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.
. . .	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
⋮	Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.
——— ⊞ ———	This symbol indicates the end of a chapter.

Contents

Illustrations

Tables

Chapter 1

An Overview of Aegis Administration

Contents

Chapter 1
An Overview of Aegis System Administration

This manual provides an overview of system administration in the Aegis environment, as well as a summary of the major changes made to the system at Software Release 10 (SR10). The manual refers to all Apollo devices that communicate on the network as "nodes," although, where appropriate, we distinguish between nodes with displays and keyboards (for example, the DN3000 series) and nodes without (for example, the DSP160) since they are sometimes configured in different ways.

Information in this manual also assumes a single Apollo network; if your site operates in an internet, please refer to the book *Managing Domain Routing and Domain/OS in an Internet* for any additional information about system administration in that environment.

This book deals with system software, that class of programs that manages the functioning of the operating system. We assume a certain level of familiarity with Aegis user-level commands and concepts. If you've read and understood the *Using Your Aegis Environment*, you should have no difficulties with anything explained in this book.

1.1 System Administration Responsibilities

As a system administrator, you are generally responsible for some or all of the following tasks:

- Enabling nodes to communicate in the network by cataloging disked nodes, providing partners for diskless nodes, and maintaining root directories.
- Creating processes to provide both network-wide and per-node services like printing, remote login, and diskless node booting.
- Understanding how to configure and administer individual nodes within the network. (While individual node users will sometimes wish to determine the system software that runs on personal nodes, the system administrator is almost always a source for advice and assistance.)
- Creating and managing a registry of authorized user and account information, including accounts and group and organization lists.
- Determining access to system software, programs, and users' files.
- Backing up both system software and user files on a regular basis.

1.2 Changes to the Operating System at SR10

We've made some major changes to broad areas of the operating system at this release. The major areas of the operating system affected, especially with regard to system administration, are the registry, protection (file and directory permissions), and the way in which the names of file system objects are represented. In the following subsections, we provide a brief discussion of the changes and their impact on Aegis system administration. Complete information about these areas, as well as procedures to perform common administrative tasks, can be found in the appropriate chapters of this book. For information on conversion tools and compatibility between pre-SR10 and SR10 versions of these subjects, see *Making the Transition to SR10 Operating System Releases.*

1.2.1 Case Sensitivity and Names

At SR10, the operating system is completely case sensitive. We do provide conversion tools, however, to ease the transition from a case-insensitive environment to a case-sensitive one. See *Making the Transition to SR10 Operating System Releases.*

The most obvious effect of case sensitivity in the operating system is that the system will interpret mixed-case pathnames literally. For example, this means that the pathnames **/DIRECTORY/FILENAME, /directory/filename,** and **/DiReCtOrY/fIlEnAmE** are no longer equivalent by default.

If you don't need to use mixed-case names, you can avoid many problems by setting the environment variable DOWNCASE to TRUE. From the point of view of naming, this will give you a pre-SR10 environment. However, DOWNCASE is intended as a temporary measure and may become obsolete at a future release. Don't rely on this mechanism for the long term.

Case sensitivity can also introduce some incompatibilities in existing shell scripts and programs. For a full discussion of the impact of the changes to naming at SR10 on programs and shell scripts, see *Making the Transition to SR10 Operating System Releases.*

The Naming Server Helper (**ns_helper**) will continue to be case insensitive, but we recommend to system administrators that new node names reflect the case-sensitive world, both to avoid confusion and because **ns_helper** may become case sensitive at a future release. The **ns_helper** is discussed in Chapter 2.

The maximum number of characters in a leaf (single file or directory) name has been increased from 32 characters to 255 characters. The maximum length of a pathname that the system can handle has increased from 256 characters to 1023 characters.

1.2.2 The Registry

The registry is the mechanism that controls user access and authentication; it has undergone significant change at SR10. Registry information is now stored in a replicated database which is managed by servers based on the Apollo Network Computing System™ (NCS).

A registry server and registry server database manage the overall registry function, and the operating system gains access to registry information via the registry server. Each node has a local registry available to provide node-specific registry history information so that a user can log in in the event of network failure. A local cache of name–to–UID (Unique Identifier) mappings is maintained on each node to improve performance.

The SR10 registry includes the concepts of membership lists, groups and organizations, which allows sites some flexibility in how the registry information is maintained. It also introduces the concept of ownership as a means of controlling access to registry database information. Simply stated, you must own a registry database relation to be able to change it. With these two additions, it is now an easy matter to partition a network's registries into logical groupings of organizations and groups, simplifying system administration.

The system administrator manipulates accounts by means of the **edrgy** tool. With **edrgy**, you can create and delete accounts, as well as edit and perform global operations on other registry database information.

It is possible to run a mixed network of pre–SR10 and SR10 machines, but you'll probably wish to locate the SR10 registries and the pre–SR10 registries on different nodes. If your network is small enough that keeping two types of registries will absorb too much disk space or be too confusing otherwise, you should consider converting to SR10 all at once. Information about operating in an environment of mixed registries is available in *Making the Transition to SR10 Operating System Releases.*

Complete information about creating and maintaining SR10 registries is available in Chapter 4. Tools are available to convert existing registries from pre–SR10 to the SR10 format and to convert SR10 registry information back to the pre–SR10 form. The intention is that, at some future release, all registries will be converted to the SR10 format. Descriptions of the various registry conversion tools and procedures can be found in *Making the Transition to SR10 Operating System Releasess.*

1.2.3 Protection: The Access Control List

At SR10, the Access Control List (ACL) mechanism, which manages file–system object protection, has been simplified and altered. Every object in the file system has an ACL which consists of four required entries for owner, group, organization, and world. Each entry consists of an SID and some rights specifications. Additional protection entries, if required, are stored in an "extended ACL" that is essentially like the pre–SR10 ACL. See Chapter 5 for information about ACLs.

As a result of these changes, so–called "canned" ACLs, predefined sets of rights specifications for certain account names, are no longer supported. The changes to ACLs also include new versions of the **acl**, **edacl**, and **salacl** commands to operate with the new and changed rights. Note that the new ACL structures will also have an impact on what protection information is preserved on backups.

Because of the general incompatibility between the SR10 ACL manager and the ACL manager prior to SR9.7, there is no easy way to share files between pre-SR9.7 and SR10 nodes. If your site has few enough nodes that you can update to SR10 all at once, you should do that; if your site is upgrading to SR10 over a long interval, and you must be able to access all files on all nodes at all times, you should install SR9.7 before updating any nodes to SR10. For information about transition and system software installation, see the books *Making the Transition to SR10 Operating System Releases* and *Installing Software with Apollo's Release and Installation Tools*.

At SR10, all mapping between UNIX modes and ACLs is handled transparently, with no intervention by the system administrator or user. The entire UNIX protection model operates exactly as you would expect on other UNIX systems.

Chapter 2

Maintaining Nodes and Providing Services in the Network

Contents

Chapter 2

Maintaining Nodes and Providing Services in the Network

This chapter assumes that your installation consists of a single Domain network. If you have multiple Domain networks connected in an internet, you have other considerations. See *Managing Domain Routing and Domain/OS in an Internet* for additional information about how to catalog nodes and maintain root directories in an internet environment.

Topics covered in this chapter include cataloging nodes and maintaining node root directories, using the **ns_helper** process to maintain root directories, and providing such network–wide services as printing and remote login.

2.1 The Root Directory

To communicate over the network, nodes must recognize each other. Each node has a root directory that associates node names and hexadecimal node IDs for all the nodes on your network; this ensures that communication and file access between and among nodes can take place. You must maintain node root directories accurately if all the nodes in your network are to operate efficiently. If a node's root directory is incomplete or inaccurate, the node may be unable to communicate with other nodes on your network.

This section describes

- Node names and node IDs
- How to catalog associations between node names and node IDs in root directories
- How to maintain root directories
- How to automate the process of maintaining nodes' root directories with the **ns_helper** (Naming Server Helper) process

2.1.1 Node IDs

Every node has a unique hexadecimal ID number (the node ID) which is contained in a Programmable Read-Only Memory (PROM). Node IDs change only if a service representative physically replaces the node's ID PROM. The node ID enables network communications software and other nodes' software to recognize that node.

Node IDs in an internet have a network number as a prefix to the hexadecimal ID that identifies the individual machine. See *Managing Domain Routing and Domain/OS in an Internet* for details.

2.1.2 Node Names

Since hexadecimal numbers are not easy to remember, Domain/OS allows you to associate a node name with a node ID. You can then refer to the node by name when using shell commands like **lvolfs** (list volume free space) that allow you to specify a node with the **-n** option. These name-ID associations are stored in the node's root directory.

A node name must begin with a letter, and all alphabetic characters in the name must be lowercase. You can assign node names to both disked and diskless nodes, but a diskless node's name does not always act the same way as a disked node's. Note the following difference between diskless and disked node names:

- A diskless nodes's name is **not** the same as its entry directory name
- A disked node's name is the same as its entry directory name

For example, if the node "dublin" were disked, the following command would list the contents of **dublin**'s entry directory.

```
$ ld //dublin
```

If the node **dublin** were diskless, the same command would result in ''object not found."

You associate node names with node IDs by means of the **ctnode** (catalog node) command. Later in this chapter, you'll find procedures that demonstrate how to catalog nodes, both in the node's own root directory and in the root directories of other nodes.

2.1.3 Cataloging a Node

The **ctnode** command enters the node's name, hexadecimal ID, and other information in the root directory. You must catalog a node whenever you

- Add a new node to the network.
- Change a cataloged node's name.
- Replace a node's disk.
- Run the **invol** utility on a node's disk.
- Have a node's ID PROM replaced. The PROM installation procedures recatalog the node's directories with its new ID. You then must update root directories on the rest of the network with the new node ID.

A new disked node arrives at your site already cataloged in its own root directory. Its default name is *node_nnnnn*, where *nnnnn* is the node's hexadecimal ID number. Diskless nodes are not already cataloged. We strongly suggest that you name all the nodes in your network.

Cataloging a node is a two–step process. First, catalog the node in its own root directory (or, for diskless nodes, in the partner's root directory). Second, make this catalog information available to all other root directories in the network. How you propagate the node name information to the other root directories on the network depends on whether your network uses the **ns_helper** server process to maintain root directories.

The ns_helper

The **ns_helper** maintains a master copy of the root directory and provides node–name to node–ID associations. It reduces the cataloging effort when you add nodes or change names, and is very useful in larger networks. You must run the **ns_helper** process if you have a Domain internet, and you should run it if your network configuration changes frequently. For complete information and procedures for using **ns_helper** and the **edns** tool which accompanies it, see Section 2.3.

The ctnode and uctnode Commands

If your network is small, node names seldom change, and new nodes are added to the network infrequently, you probably don't need to run **ns_helper**. In this case, you'll use the **ctnode** and **uctnode** (uncatalog node) commands and Procedures 2–1 through 2–6 in this chapter to maintain the root directories of nodes on your network. Many of these procedures will not operate in an internet. See *Managing Domain Routing and Domain/OS in an Internet* for information about using **ns_helper** on a Domain internet.

2.1.4 Cataloging a Node in its Own Root Directory

Use Procedures 2–1 and 2–2 to catalog a disked or diskless node in its local root directory. The procedures catalog the diskless node in its partner node's root directory. Use one of these procedures whenever you catalog a node except for when a PROM is changed. In this case, the PROM installation procedures recatalog the node in its own root directory; however, you must still recatalog the node in other nodes' root directories if you do not use **ns_helper**.

- Use Procedure 2–1 to catalog a node that has a display (that is, any node except a Domain Server Processor (DSP).
- Use Procedure 2–2 for a a server node that does not have a display.
- If you are cataloging more than one node, use Procedure 2–1 or 2–2 at each node you are cataloging.
- These procedures only catalog a node in its local root directory. If you do not use **ns_helper**, you must continue with Procedure 2–3, 2–4, 2–5, or 2–6 to provide this information to all other nodes.

Please read through each procedure before you attempt to carry it out. If you receive error messages when you carry out the procedures, check the command line to be certain that you have given the correct input. If you are sure you are giving the correct input but you continue to receive error messages, check with Customer Service or your designated service representative.

Procedure 2-1. *Cataloging a Node in its Own Root Directory*

Task 1: Log in as user

Task 2: Determine the node's hexadecimal ID

```
$ netstat
The net_ID.node_ID of this node is 29C27.4DD0.
```

Task 3: Uncatalog the old node name

If this is a new diskless node, you replaced the disk on an already-cataloged node, or you ran **invol**, go to Task 4. If this is a new disked node, the initial node name is the node ID preceded by **node_**, for example, **node_8523**. In the following example, the **–l** option lists the node's name after it is uncataloged. Note that you do not use // before the node name.

```
$ uctnode node_8523 -l
 "node_8523" uncataloged.
```

Task 4: Catalog the new node name

Enter the following command if you are cataloging a new node or are giving a node a name that has never been used before. For example, to name the node with hexadecimal ID 8523 "salmo," type the following:

```
$ ctnode salmo 8523 -l
  Node 8523 cataloged as "salmo".
```

Enter the following command if you are reusing an existing name. You usually reuse a name when you change disks, run **invol**, or replace a node and the user wishes to keep the old node name for the new node.

$ **ctnode** *old_name node_id* **–l –r**

Task 5: Update the node's root directory

This step adds node name-ID associations for other nodes on the network to this node's master root directory.

```
$ ctnode -update -l
 3 nodes responded!
 Node 8555 cataloged as "arctic_char"
 Node 8523 cataloged as "rainbow"
 Node 8525 cataloged as "brook"
```

Procedure 2–2. *Cataloging a Domain Server Processor*

Use this procedure to catalog DSPs. If you're setting up a new network, catalog DSPs after you've cataloged nodes with monitors. Get the DSP's node ID from the inspection slip attached to the shipping carton packing slip. If you do not have the inspection slip, contact your service representative; this is the only reliable way to determine the node ID when the node is uncataloged and you don't have the packing slip. You must have the node ID before you start this procedure.

Task 1: Log in to the DSP as user

Enter the following command at a shell prompt on a node with a monitor. Note the two single quotes ('') at the end of the command line, which show that the account user has a null password. For example, if the DSP's node ID is 8533, login as user and type the following:

```
$ crp -on 8533
Connected to node 8533
```

Task 2: Uncatalog the old node name

If you replaced the disk on an already-cataloged node, or you ran **invol**, go to Task 3. If this is a new DSP, the initial node name is the node ID preceded by **node_**, for example, **node_8533**. In the following example, the **–l** option lists the node's name after it is uncataloged.

```
$ uctnode node_8533 -l
 "node_8533" uncataloged.
```

Task 3: Catalog the new node name

Enter the following command if you are cataloging a new DSP or are giving a DSP a name that has never been used before. For example, type the following to associate the name "chinook" with the DSP with node ID 8533.

```
$ ctnode chinook 8533 -l
Node 8533 cataloged as "chinook".
```

Enter the following command if you are reusing an existing name. You usually reuse a name when you change disks, run **invol**, or replace a node and the user wishes to keep the old node name for the new node.

```
$ ctnode old_name node_id -l -r
```

Task 4: Update the node's root directory

This step adds node name–ID associations for other nodes on the network to this node's master root directory.

```
$ ctnode -update -l
 3 nodes responded!
Node 8523 cataloged as "rainbow"
Node 8525 cataloged as "brook"
Node 8533 cataloged as "chinook"
```

2.2 Using ctnode to Catalog Nodes on the Network

Once you catalog a node in its own root directory, you must then provide the information to all other nodes' root directories, so that remote nodes can communicate with the newly cataloged node and have access to its files. If the network is small and node configurations don't change often, you can use the **ctnode** and **uctnode** commands to manage the network root directories. Procedures 2–3 through 2–6 show the steps you must follow to update the root directories. Use these procedures as detailed below. If you have a larger network, you should probably run the **ns_helper** process. Go to Section 2.3 for information and procedures for using **ns_helper**.

- Use Procedure 2–3 to create a new network or to add several nodes to a network.
- Use Procedure 2–4 to add a single node to an existing network.
- Use Procedure 2–5 to change the name of a node that is already on the network.
- Use Procedure 2–6 after replacing a disk drive, running **invol**, or if your service representative replaced a node's PROM.

All these procedures assume that you've already cataloged the node in its own root directory, using either Procedure 2–1 or 2–2.

Procedure 2–3. *Creating a New Network*

Task 1: **Log in as user**

Task 2: **Update the root directory**

Enter the **ctnode –update** command to update the node's root directory to include information on all nodes that are currently responding to network queries. In the following example, the **–l** option lists the nodes as they are cataloged.

```
$ ctnode -update -l
2 nodes responded!
Node 8555 cataloged as "arctic_char"
Node 8525 cataloged as "brook"
```

The local node now has a complete root directory. If the number of nodes responding does not equal the number of nodes in your network, repeat Task 2 until you get a full root directory.

Task 3: **Propagate new information across the network**

You must propagate the new name–ID information to the root directories of all other nodes. Enter the name of the node you're logged into in place of *//node_name* in the following command line:

```
$ ctnode -md -from //node_name -on //?*
```

Procedure 2–4. *Cataloging a New Node in an Existing Network*

Task 1: **Log in as user**

Task 2: **Catalog the new node**

Catalog the new node on all other nodes in the network with the following command. Substitute the node's name and ID in the appropriate places.

```
$ ctnode node_name node_ID -on //?*
```

Procedure 2–5. *Changing a Node's Name*

Task 1: **Uncatalog the old name**

Repeat this task at each node on the network to uncatalog the node's old name. Otherwise, the node will be cataloged under both the new and the old name.

Log in as **user** and enter the **uctnode** command to remove the node's old name from the root directory. In the following example, the node's old name is "cutthroat."

```
$ uctnode cutthroat -l
 "cutthroat" uncataloged.
```

Task 2: **If you are not still logged in, log in to any node as user**

Task 3: **Update root directories**

To propagate the new node name to the root directories of all nodes in the network, recatalog the node under its new name. In this example, the node ID is **cff.**

```
$ ctnode sockeye cff -r -on //?*
```

Procedure 2–6. *Updating Information for an Existing Node Name*

Use this procedure after replacing a disk drive, running **invol**, or if your service representative replaces a node's PROM.

Task 1: **Log in to any node as user**

Task 2: **Recatalog the node in its own root directory**

To recatalog the node in its own root directory, substitute the node's name and ID in the following command:

$ **ctnode** *node_name node_ID* **–r**

Task 3: **Update the root directories across the network**

To propagate the updated information into the root directories of all nodes in the network, enter the recataloged node's name and ID in the following command:

$ **ctnode** *node_name node_ID* **–r –on //?***

2.3 The Naming Server Helper: ns_helper

The **ns_helper**, the Naming Server Helper, is a Domain server process that provides an automated method of maintaining node root directories. You can run **ns_helper** on any disked nodes in a Domain network. You must run it on each Domain network in an internet. See *Managing Domain Routing and Domain/OS in an Internet* for information about running **ns_helper** in an internet.

The **ns_helper** (**/sys/ns/ns_helper**) process manages a master root directory. This database is the only comprehensive source of node-identifying information in the network. The **ns_helper** performs most of its operations automatically. The **edns** utility, an interactive tool used with **ns_helper**, is available for those operations requiring your intervention, such as updating the database.

The **ns_helper** maintains a cache of the master network root directory at each node. Whenever the naming server uses the master root directory to locate objects, it updates the local node's cache. Although the shell command **ctnode** is operative, you need not maintain a node's root directory with **ctnode -update** in the **ns_helper** environment. It is always necessary to catalog an entry directory name with **ctnode** when a node is first brought into the network.

When more than one **ns_helper** runs in a network, each process is called a replica. The **ns_helper** propagates changes in the database of any replica to all other replicas. If **ns_helper** is not successful in propagating changes, it will continue to try for a period of 14 days. In exceptional circumstances of node, loop, or disk failure, a replica may not receive updated information in this time period. Use an **edns** merge command to return replicated databases to a consistent state in these cases.

We recommend running **ns_helper** as a background process. Enable **ns_helper** from the appropriate start-up file (usually **/etc/rc**) so that it will continue after logout. The **ns_helper** names itself "ns_helper" by default, so you need not specify the **-n** option to the process creation command.

2.3.1 The ns_helper Database

The **ns_helper** manages a database that is divided into two parts: a master root directory and a replica list. The master root directory is the comprehensive source of node identification information in the network. You can specify the node names and addresses in the master root directory. Only the nodes themselves can supply **ns_helper** with the rest of the information in the directory. The **ns_helper** database resides in the **'node_data/system_logs** directory.

On large networks and on Domain internets you can have more than one **ns_helper** process, each with its own copy of the database; these are called replicated **ns_helpers** and databases. In this case, the database replica list includes the nodes that run **ns_helper**.

Table 2-1 lists the **ns_helper** database contents in detail.

Table 2-1. Contents of the ns_helper Database

Item	**Definition**
	Master Root Directory
Node Name	The name of the node.
Address	The network number (0 for Domain single networks) and the node's hexadecimal ID.
Entry Type	The type of object — for disked nodes, system directory (sdir); for diskless nodes, node (node)
UID	The Unique Identifier (UID) for a node's entry directory For diskless nodes, **ns_helper** assigns a UID.
Entry Date and Time	The date and time at which this entry was added to the master root directory.
Creating Node	The hexadecimal ID of the node at which the entry was added to the master root directory.
	Replica List
Replica List	The hexadecimal IDs of all nodes that run the **ns_helper** process.

2.3.2 When to Use ns_helper in Your Network

Use **ns_helper** in networks where new nodes are introduced frequently, when many nodes have multiple users, and when you want to maintain all the node root directories from one location. As we said before, you must run **ns_helper** on each network in an internet.

2.3.3 Number and Placement of Replicated ns_helpers

In smaller networks, a single **ns_helper** process provides a reliable way to keep nodes up to date. In some environments, however, you may choose to run the **ns_helper** server on several nodes. Consider running more than one **ns_helper** process when

- You have many nodes in your network, and a single server may not be able to handle the traffic.
- You want to ensure that the server process is always available; two or more servers will provide redundancy.
- Loops in your network are switched out regularly and/or your network runs through several buildings. You might want servers in each loop or building.

You can run **ns_helper** only on disked nodes.

2.3.4 Replica List

When more than one **ns_helper** runs on a network, the server processes maintain information about each other in a part of their database called the replica list (see Table 2–1). The replica list contains the hexadecimal ID of every node running **ns_helper**. You manage replicated **ns_helpers** with the **edns** command.

Each **ns_helper** automatically tries to keep its own database (master root directory and replica list) consistent with those of the other **ns_helpers**. When you make changes to any database, that node's **ns_helper** refers to its replica list for the node IDs of other replicas. Then the **ns_helper** sends the new information to all the other nodes on the list. When **ns_helpers** receive new information from another server, they update their own databases and return an acknowledgment to the sending server.

If the sending **ns_helper** does not receive an acknowledgment from all the nodes on its replica list, it continues to propagate the new information for 14 days. In exceptional circumstances, a replica might never receive updated information. You can use the **edns merge** facility to return replica databases to a consistent state in these cases.

2.3.5 Managing Root Directories with ns_helper

When **ns_helper** runs in a network, the naming server (the part of the operating system that locates file–system objects) has two sources of information about entry directory names: the node's local root directory and the **ns_helper** master root directory itself.

When the naming server tries to locate an object, it first looks in the node's root directory. If the name isn't there, the naming server refers to **ns_helper**'s master root directory for information about the entry name. Whenever the naming server gets information from the master root directory, it adds that information to the node's root directory.

Using ctnode and uctnode with ns_helper

When you use **ns_helper** on the network, you normally will not need to use the **ctnode** command to maintain a node's root directory.

There is one situation in which you'll need to use **uctnode**: when you change the name of a node in the **ns_helper** database, the new name is added to the individual nodes' root directories, but the old name is not deleted from any of them. Use **uctnode** at each node on the network to remove the old entry from all root directories.

> NOTE: After **uctnode** removes an entry directory name, objects cataloged under that node's entry directory are no longer accessible to you or other nodes on the network.

The following subsections describes tools, considerations, and procedures for managing root directories on networks that use **ns_helper**.

The edns Utility

The **edns** (edit naming server) utility manages the **ns_helper** and its database. Table 2-2 briefly describes the **edns** commands. The *Aegis Command Reference* describes the **edns** commands in greater detail. Online help is also available by using the **help** command.

Synchronizing Clocks on Replicated Databases

The **ns_helper** processes keep only the most recent information about an entry in their databases. The servers use the node hardware clocks and the database item "Entry Date/Time" to recognize the most recent information. Therefore, you must keep the hardware clocks on all **ns_helper** nodes synchronized. Check the node clocks periodically and reset them if they diverge by more than a few minutes. The **edns** command provides a way to check clock synchronization (see Procedure 2-15). (Procedure 2-7 explains how to synchronize node clocks.)

Network Availability and edns

Two **edns** operations, **initialize** and **update**, are particularly sensitive to network and node availability because they request information from all nodes on the network. If a node fails to respond, it may not be cataloged in the root directory. Therefore, it is a good idea to initialize the first **ns_helper** process at a time when network traffic is light and all nodes are connected to the network.

Table 2-2. The edns Commands

Command	Description
add	Adds a node name and the corresponding address to the master root directory(s).
addrep	Adds the address of an **ns_helper** node to the **ns_helper** replica lists(s).
cmp	Compares two **ns_helper** databases and lists entries that appear in both, or that appear inconsistently in both.
delete	Deletes the entry for the specified name from the **ns_helper** master root directory(s).
delrep	Deletes an **ns_helper** node from the **ns_helper** replica lists.
diff	Lists the differences between two **ns_helper** databases, including both the master root directories and replica lists.
info	Displays address and status information for the default **ns_helper**.
init	Initializes an **ns_helper** database with data from all nodes that are currently responding on the network.
ld	Lists master root directory information.
lr	Lists the addresses of all **ns_helper** nodes on the network; can display the nodes' current clock dates and times.
merge	Merges all entries from one **ns_helper** master root directory (but not replica list) into another.
merge_all	Performs a global merge of all **ns_helper** databases, using the default or specified **ns_helper** database.
quit	Ends the current **edns** session.
replace	Changes the address and UID associated with the specified name.
set	Sets the default **ns_helper** to be the one running on the specified node.
shut	Shuts down the **ns_helper** on the specified node. This command deletes that node's database but does not remove the node from other **ns_helper** replica lists.
update	Updates all replica master root directories with data from all nodes that are currently responding on the local network.

The edns Utility and Diskless Nodes

When **edns** initializes a database, it always assigns the default name

diskless_$*nnnnnn*

to a diskless node, where *nnnnnn* represents the diskless node's hexadecimal ID. The value is right justified and is preceded by the number of zeros required to form a 6-digit number. For example, if the node ID is 3d3, the **edns** representation of that node is

diskless_$0003d3

Edns generates a Unique Identifier (UID) for a diskless node, then associates it with the diskless node's name. This information about the diskless node appears in the master root directory and can be copied to a node's root directory. In a single Domain network, the UID and other information that **edns** generates for the diskless node serves only as a place marker for information that is used in a Domain internet.

To name a new diskless node on an existing network that runs an **ns_helper** process, use the **ctnode -root** command described in the next section, or use Procedure 2-10 described in Subsection 2.3.7. If the node was on the network when **ns_helper** was initialized, it already has a default name; in this case, use Procedure 2-12 or use **uctnode -root** to uncatalog the node before recataloging it.

The command **ld // -ln** lists the names of all diskless nodes in the local copy of the root directory. The following commands also specify if a node is diskless. (The *node_spec* argument can be either a hexadecimal node ID or the node entry directory, that is, *//node_name*.)

```
$ netstat -n node_spec
$ lusr -n node_spec
$ bldt -n node_spec
$ lcnode -me
```

2.3.6 User Procedures for Updating the Master Root Directory

The **ctnode, uctnode,** and **ld** commands support a small subset of operations on the master root directory. Any user can run these commands to manage the master root directory entries.

Use the following command to delete the entry for a node name in both the node's root directory and the master root directory. If you remove a node from the network, this command removes the old node's entry from the master root directory. If you are changing a node's name, use this command to remove the entry for the old name before adding the new name.

```
$ uctnode node_name -root
```

> **NOTE:** Any time you change a node's name, you must use **uctnode** on each node to delete the old entry from that node's root directory.

Use the following command to add a node name in the root directory and the master root directory. You can use this command to give a diskless node a name or to add a new node to the network.

```
$ ctnode node_name node_id -root
```

Use the following command to replace the node ID or UID that is associated with an existing node name in the root directory and the master root directory. You can use this command if your disk is changed or if you run **invol**. You can also use this command to associate an existing name to a new node.

```
$ ctnode node_name node_id -root -r
```

You can list the contents of the master root directory by typing

```
$ ld -root
```

2.3.7 System Administrator Procedures for ns_helper

The rest of this subsection contains procedures for managing **ns_helper** processes and databases. Table 2–3 lists tasks you can perform and gives the number of the appropriate procedure. In secure networks, we recommend that you set the ACLs so that only the system administrator (%.sys_admin.%) can use the **edns** command. If you do, only the system administrator can perform Procedures 2–9 through 2–16 that invoke **edns**. Other users must run the **ctnode** and **uctnode** commands to manage a node's entry in the master root directory as described in the preceding section.

Table 2-3. The ns_helper Procedures

Description of Procedure	**Number**
Add a node to an existing network	2-10 *
Add node names to the **ns_helper** database	2-10 *
Add an **ns_helper** replica	2-14
Change a node's name in the **ns_helper** database	2-12 *
Check clock synchronization on **ns_helper** nodes	2-15
Delete names from the **ns_helper** database	2-11 *
Give a diskless node a name	2-10, 2-12 *
Initialize a network's **ns_helper** database	2-9
Maintain the consistency of replicated **ns_helper** databases	2-16
Reinitialize a single **ns_helper** process	2-14
Remove an **ns_helper** replica	2-17
Remove a node from the network	2-11 *
Repair a replica	2-16
Start **ns_helper** on one or more nodes	2-8
Stop an **ns_helper** process	2-18
Synchronize node clocks	2-7
Update **ns_helper** after changing a PROM or running **invol**	2-13 *
NOTE: An asterisk (*) indicates a procedure that you can also perform by using **ctnode** and **uctnode** commands.	

Procedure 2–7. *Synchronizing Node Hardware Clocks*

Use this procedure to synchronize node hardware clocks. You must use this procedure if nodes that run **ns_helper** are not within five minutes of each other. You should use the procedure if the **ns_helper** nodes are not within one minute of each other.

Task 1: Obtain an accurate timepiece

Task 2: Set the node's NORMAL/SERVICE switch to SERVICE

Task 3: Shut down the system

If the node is a DSP unit without a display, you must attach either a terminal or a Domain node with a display to the DSP unit's SIO (Serial Input/Output) line to set the node clock. Once you've done that, shut down the DSP.

If the node has a display, shut it down with the DM (Display Manager) **shut** command.

Task 4: Start the calendar program

The Mnemonic Debugger prompt (>) appears on the screen. Enter the following command:

```
> ex calendar
```

The **calendar** program prompts you for the required responses. (You must answer all questions; you cannot set the time without also entering the date.) For example, the following script illustrates prompts from **calendar**, and the responses for a node with a Winchester disk:

```
> ex calendar
Please enter disk type (W,S, or F)[,lvno].
If you do not have a disk, enter none (N): w

The time-zone is set to -4:00(EDT).
Would you like to reset it? n

The calendar date/time is 1988/07/11 13:52:03 EDT.
Would you like to reset it? y

Please enter today's date(year/month/day): 1988/07/11
Please enter the local time in 24 hr. format (hours minutes) 13:55

The calendar has been set to 1986/07/11 13:55 EST
(1986/07/11 18:55:04 UTC)
```

NOTE: If you set a node clock backward, the node can generate duplicate UIDs for objects, which will create confusion in the operating system. You can avoid this by not rebooting until the amount of time you set the node clock past has elapsed. For example, if you set the clock back by one hour, wait one hour before you reboot the node.

Task 5: Reboot the node or DSP

Return the NORMAL/SERVICE switch to the NORMAL position and reboot. Type

```
> re
> Press <RETURN> twice
> ex domain_os
```

Task 6: Repeat Tasks 2 through 5

Using the same timepiece, repeat Tasks 2 through 5 at each of the nodes you selected to run **ns_helper**.

Procedure 2–8. *Starting the ns_helper Server Process*

Use this procedure to start or restart the **ns_helper** on any node that maintains a master root directory.

Task 1: **Check node clock synchronization**

If more than one node runs (or will run) **ns_helpers**, check to make sure that the nodes' clocks are within one minute of each other. To do so, enter the **netstat –n** command, followed by the node specifications of the nodes that run **ns_helpers**. In this example, the nodes **brook** and **golden** run **ns_helper**.

```
$ netstat -n //brook //golden
     The net_id.node_ID of this node is 0.5678.
**** Node 4567 ****  //brook
Time 1986/07/01.15:25:57  Up since 1986/07/01.14:38:25
Net I/O:          total=  24555   rcvs =  17930   xmits =   6625
Winchester I/O:   total=  13837   reads=  11098   writes=   2739
No ring hardware failure report.
System configured with 3.0 mb of memory.
**** Node 1345 ****  //golden
Time 1986/07/01.15:21:44  Up since 1986/06/24.20:57:25
Net I/O:          total= 5572914   rcvs = 3892617   xmits = 1680297
Winchester I/O:   total= 244976   reads= 148445   writes=  96531
System configured with 2.5 mb of memory.
```

If the times reported are not within one minute of each other, use Procedure 2–7 to synchronize the hardware clocks on the **ns_helper** nodes.

Task 2: **Log in**

Log in to the node that will run **ns_helper**. Use the appropriate procedures for nodes with monitors or DSPs.

Task 3: **Edit the start-up file**

Add the following line to the appropriate node start-up file in /sys/**node_data**:

```
    cps /sys/ns/ns_helper
```

This line will start an **ns_helper** process on this node, when the node is rebooted.

Task 4: **Start the ns_helper process**

If you are working at the node you want to run **ns_helper**, enter the following command in the DM input window:

```
Command: cps /sys/ns/ns_helper
```

If you are working at another node for example, if the **ns_helper** node is a DSP server, enter the following command. You must use this command even if you are logged in remotely to the **ns_helper** node.

$ **crp –on** *node_spec* **–cps /sys/ns/ns_helper**

where *node_spec* is the node name.

Task 5: Verify that the server process is running

Enter the **pst** command and specify a node name. The output from the command is shown below.

```
$ pst -n //node_name
------------------------------------------------------------
Processor   | PRIORITY   |  Program | State |   Process
Time (sec)  | mn/cu/mx   |  Counter |       |   Name
------------------------------------------------------------
 70.938       16/16/16    1BDE6      Wait     display_manager
 21.297       1/14/16     <active>   Ready    process_7
  0.850       1/14/16     1BD46      Wait     ns_helper
.
.
.
```

Task 6: Repeat Tasks 2 through 5

Repeat Tasks 2 through 5 at each of the nodes you selected to run **ns_helper**.

Procedure 2–9. *Initializing the Network ns_helper Database*

Use this procedure to initialize the **ns_helper** database on a new Domain network. We illustrate the following procedure with **ns_helper** running on two nodes: **golden** (ID 8521), and **brook** (ID 8525), and with six nodes in the network: **golden, brook, grayling, gila, arctic_char,** and **coho.**

Task 1: Catalog all nodes in their own root directories

Be sure that all nodes in the network have their own entry directory names cataloged in their own root directories. Procedures 2–1 and 2–2 show how to do this.

Task 2: Start ns_helper on each helper node

Use Procedure 2–8 to start the **ns_helper** process on the nodes you've selected.

Task 3: Start edns

Invoke **edns** from any node in the network with the node specification of the node that will run **ns_helper**. Our example selects **brook**; its **ns_helper** process becomes the default **ns_helper**.

```
$ edns 8525
the default ns_helper is 0.8525
<edns>
```

In the display, <edns> is the **edns** prompt.

Task 4: Initialize the ns_helper database

Use the **edns init** command to initialize the **ns_helper** database that will reside on this node.

```
<edns> init
6 nodes responded to lcnode request
6 entries added to directory
0 names already existed 0 errors
```

Task 5: Verify that all nodes are in the database

Enter the **edns ld** command to list the database and verify that it includes all nodes on the network.

```
<edns> ld
grayling  brook       golden
gila      arctic_char coho
6 entries listed.
```

Repeat Task 3 if some nodes were not added to the directory. Skip to Task 10 if you are only going to have one **ns_helper** node running on the network.

Task 6: Create an ns_helper replica

If you wish to run more than one **ns_helper** process, use the following command to set the default server process to a replica **ns_helper** node. In our example, we're setting the default server to the process running on **golden**, so that we can create an **ns_helper** database replica on that node.

```
<edns> set 8521
The default ns_helper is 0.8521
```

Task 7: Initialize the replica database

You must initialize the replia database (in the example, **golden**) with information from the original database (on **brook**).

```
<edns> init -from 8525
```

Task 8: Verify that the two databases are identical

Use the **edns diff** command to verify that the original and replica **ns_helper** databases contain the same information.

```
<edns> diff golden brook
The two directories are identical
The two replica lists are identical
```

If the databases are not consistent, repeat Task 7.

Task 9: Create additional replicas

Repeat Tasks 6 through 8 for each additional replica node that you are initializing.

Task 10: Quit the edns program

End the **edns** session by typing the following:

```
<edns> quit
$
```

Procedure 2–10. *Adding Nodes to the ns_helper Master Root Directory*

Use this procedure to add a node to the network and to name a diskless node if the node was not on the network when the **ns_helper** database was initialized or updated. The procedure updates the **ns_helper** master root directory with the information for the new node and propagates the information to all replica databases. In the example, the nodes added to the master root directory are **laker, paiute, brown,** and **dolly_varden.**

Task 1: Invoke edns

From any node, enter the **edns** command.

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Add nodes to the default ns_helper root directory

Use the following command to add each new node's name and ID to the default **ns_helper**'s root directory:

<edns> **add** *node_name node_id*

For example:

```
<edns> add laker 7206
<edns> add paiute 128c
<edns> add brown 3333
<edns> add dolly_varden 4b70
```

Task 3: List the root directory

Changes to the database take effect immediately. List the directory to be certain you did not make errors.

<edns> **ld -n**

```
nodeid name
7206   laker
3333   brown
4b70   dolly_varden
128c   paiute
503f   grayling
8525   brook
8521   golden
1c68   gila
 5f6   arctic_char
70de   coho

10 entries listed.
```

Use the **edns del** command to remove any errors you have made; then use the **add** command to correct information.

Procedure 2–11. *Deleting Names from the Master Root Directory*

Use this procedure to remove a node from the network or to delete any entries that you added incorrectly to the master root directory. In this example, the nodes **laker** and **paiute** are deleted from the master root directory.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Delete entries

Use the following command to delete the entries that you want to remove:

```
<edns> del laker
<edns> del paiute
```

Task 3: List the root directory

Changes to the database take effect immediately. List the directory to be certain you did not make errors.

```
<edns> ld
brown        dolly_varden        grayling
brook        golden              gila
arctic_char                      coho

8 entries listed.
```

Procedure 2–12. *Changing a Node's Name*

Use this procedure if you change a node's name, for example if you change the name of node 3333 from "sunapee" to "speckled." Also use this procedure to give a diskless node a name if the node was on the network when the **ns_helper** database was initialized or updated. (In this case, the diskless node already has the default name **diskless_$***nnnnnn*, where *nnnnnn* is the node ID.)

Task 1: **Uncatalog the old node name**

Repeat this task at each node on the network. Otherwise, the node will be cataloged under both the new and the old name. Log in as user. Use the **uctnode** command to remove the node's old name from the root directory.

```
$ uctnode sunapee -l
 "sunapee" uncataloged.
```

Task 2: **If you are not logged in, log in at any node**

Task 3: **Invoke edns**

From any node, enter the **edns** command.

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 4: **Delete the node's old entry from the master root directory**

Use the **edns del** command to delete the node's old entry from the database.

```
<edns> del sunapee
<edns> del diskless_$009a7d
```

Task 5: **Add new entries**

Use the **add** command to create a new entry in the database with the node's new name.

```
<edns> add speckled 3333
<edns> add nodisk 9a7d
```

Procedure 2–13. *Replacing a Node's Hex ID for a Node in the Master Root Directory*

Use this procedure whenever you replace a node's disk with another disk and whenever you use the **invol** utility. These operations replace the node's ID, and you must put the new information in the database.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Replace the old information

Use the following command to replace the node ID information for the node named **coho.**

```
<edns> rep coho 70de
```

Procedure 2–14. *Adding or Initializing an ns_helper Replica*

Use this procedure when you add an **ns_helper** replica. Use Tasks 2 through 4 to reinitialize the database of an existing **ns_helper** replica. In this procedure's examples, we initialize the new **ns_helper** replica on node **coho,** node ID 70de, from **brook,** node ID 8525.

Task 1: Start ns_helper on the new node

Use Procedure 2–8 to start **ns_helper** on **coho.**

Task 2: Invoke ns_helper and set the new default

At any node, enter the following command to invoke **edns** and set the default **ns_helper** directory to the new replica node, **coho.**

```
$ edns 70de
The default ns_helper is 0.70de
```

Task 3: Initialize the replica database

Use the following **edns** command to initialize the replica database from the existing **ns_helper** replica at the node **brook.**

```
<edns> init -from 8525
```

Task 4: Verify master root directories

Use the following **edns** command to verify that the master root directory and replica list are the same on both the original replica node and the new replica node.

```
<edns> diff coho brook
The two directories are identical
The two replica lists are identical
```

If the databases are not consistent, repeat Task 3.

Procedure 2–15. *Checking Replica Node Clocks*

You must keep the clocks on **ns_helper** nodes synchronized because the **ns_helper** applies time stamps to the information in its database. Skewed clocks can result in new data from an **ns_helper** node with a slow clock being deleted and replaced by older (and inaccurate) data from a replica node with a fast clock. We recommend that you keep replica node clocks within one minute. The ns_**helper** allows more than one minute of divergence and will indicate the skew after the range exceeds five minutes.

You should check the replica nodes' clocks daily until you know how long it takes for the clocks to diverge. You can then adjust the replica nodes' clock inspection schedule accordingly.

To check the **ns_helper** node clocks, perform the following tasks:

Task 1: Invoke edns

From any node, enter the **edns** command:

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 2: List replicas' clock times

Enter the following **edns** command.

```
<edns> lr -clocks
```

The following message from **lr** verifies that the replica nodes' clocks are synchronized well enough for **ns_helper** to operate properly:

```
 replica   datetime
 0.0070de  86/08/09.16:52
 0.008525  86/08/09.16:53
 0.008521  86/08/09.16:54
All clocks are synchronized to within ns_helper threshold.
```

The next message indicates that clocks are skewed:

```
 replica   datetime
 0.0070de  86/08/09.16:51 clock skewed
 0.008525  86/08/09.16:58 clock skew warning
 0.008521  86/08/09.17:03 clock skewed
```

Task 3: Synchronize skewed clocks

If the clocks are skewed, use Procedure 2–7 to synchronize them. Then use Procedure 2–16 to check the replica databases for inconsistencies and, if necessary, make them consistent.

Procedure 2–16. *Maintaining Consistency of Replicated Databases*

The example used in this procedure compares and unifies the databases on the **golden** and **brook** nodes.

Task 1: Compare Replica Databases

1. Use the **edns diff** command to check whether two replica databases are the same. This command shows any discrepancies in names, UIDs, or addresses. For example, it reports any discrepancies that might have been caused by skewed clocks. It also reports any differences in the replica lists.

   ```
   <edns> diff golden brook

                value in          value in
              0.008521         0.008525
   diff           directory          directory              name

   name          name found     name not found       gila
    uid   21089D87.4000503f  2108af41.4000503f      grayling

    The two replica lists are identical
   ```

   In this example, **diff** shows that **gila** is cataloged in **golden**'s master root directory but not in **brook**'s. It also shows that **grayling** has a different UID value in the two replica master root directories.

2. Repeat the **diff** command until you have compared all replica nodes. For example, if **ns_helper** runs on **golden, brook,** and **coho,** the following two commands compare all replicas:

   ```
   <edns> diff golden brook
   <edns> diff golden coho
   ```

Task 2: Unify Replica Databases

The **edns** utility provides two commands for making replica databases consistent: **merge** and **merge_all.**

The **edns merge** command updates one replica database from a second database. It operates on both the directory and replica list of an **ns_helper**. In the following example, **merge** adds to **brook**'s database any information that is present in **golden**'s database but absent from **brook**'s. It also replaces any information in **brook**'s database that is older than information about the same entry in **golden**'s database. It is important to note that nothing is changed in the **–from** replica (that is, **golden**'s) database. Type

```
<edns> merge brook –from golden
```

Since **merge** only operates on the target replica database, it is often appropriate to merge all replicas into a consistent state. The **edns merge_all** command merges all replica databases, using either the default replica node or a specified node as the source node. The command merges the information from all **ns_helper** replicas on the source node's replica list into the source node's database, using the most recent information whenever there are conflicts. It then updates all other replica databases with the contents of the source node's database. However, it can only get information from and update **ns_helpers** that are on its replica list. Therefore, check to make sure that the source node's replica list includes all replica nodes before you use **merge_all**. The following steps use **merge_all** to ensure that all databases are consistent.

1. Invoke the **edns** utility:

   ```
   $ edns
   The default ns_helper is 0.8521
   ```

2. Enter the **lr** command to list names of the nodes in the default **ns_helper's** replica list:

   ```
   <edns> lr
   replica
   0.008525
   0.008521
   ```

3. If any **ns_helper** replica nodes are missing from the list, use the **addrep** command to add the node. For example, the replica list in Step 2 is missing **coho** (node ID 70de). Enter the following command to add the node:

   ```
   <edns> addrep coho
   ```

4. Merge all replica databases by entering

   ```
   <edns> merge_all
   ```

In this case, the default replica node (**golden**) is the source node, and this command merges the information from **brook** and **coho** into **golden.** It then merges **golden** into **brook** and **coho.**

Procedure 2–17. *Removing an ns_helper Replica*

Use this procedure to stop an existing **ns_helper** replica process, delete the node's replica database, and delete the node's name from all other **ns_helper** nodes' replica lists.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Delete the replica

Use the **edns delrep** command to delete the replica.

```
<edns> delrep coho
```

The **delrep** command deletes the specified node's ID from the replica list of all other **ns_helpers**. It also causes the **ns_helper** at the specified node to delete its database and stop active service to the network. The inactive replica does not accept new transactions and will only accept **edns info** requests. It continues to run until it has completed sending any information it has that has not yet been propagated to the other replica nodes. When all information has been sent, it stops running.

Task 3: Check for running server

Use the **pst** command to see if the server process is still running from time to time. If you are at the replica node, enter

```
$ pst
```

If you are at any other node, enter

```
$ pst -n //replica_node_name
```

where *node_name* is **coho**.

If **ns_helper** is not mentioned in the process list, it has stopped running.

If the deleted **ns_helper** is still running after a few days, you may have to stop it manually. This is the case if the deleted **ns_helper** has stopped, but the node is rebooted before you do Task 4; the stopped **ns_helper** restarts when the node reboots. Use the **edns info** command to see if the **ns_helper** can now be stopped.

```
$ edns 70de
The default ns_helper is 0.70de
<edns> info
The default ns_helper is 0.70de
Its status is uninitialized.
```

If the **info** command reports that the deleted replica **ns_helper** is uninitialized, then it is appropriate to stop it.

```
<edns> shut 70de
```

Task 4: Disable ns_helper startup

When the process has stopped, remove the following line from the appropriate node start–up file on node 70de (**coho**).

```
cps /sys/ns/ns_helper
```

This prevents the **ns_helper** process from retstarting on **coho** the next time the node is rebooted.

Procedure 2–18. *Shutting Down an ns_helper Replica*

Use this procedure to immediately shut down an **ns_helper** and delete its database without deleting the replica node ID from other **ns_helper**'s replica lists. Any information that the **ns_helper** has not yet sent to other replicas will not be sent and the information may be lost. Therefore, you should use Procedure 2–16 before you delete this replica.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
$ edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Shut down the replica

```
<edns> shut coho
```

2.4 Aegis Mail Services

Domain Professional Support Services (DPSS™) Mail is the mail system for Aegis users. This subsection briefly describes DPSS/Mail™. For more detailed information on using and administering DPSS/Mail, see the *DPSS/Mail User's Guide*.

DPSS/Mail uses a subscriber directory to identify mail addresses. This directory is part of the registry. You must ensure that the directory contains each mail user's mail address (using the **edsd** command).

DPSS/Mail can be used to send mail to UNIX systems, Sendmail and Alis*.

2.4.1 DPSS/Mail and UNIX Mail

A UNIX mail gateway, supplied with DPSS/Mail, can be accessed by sending mail to addresses of the form *user@unix* (for DPSS to UNIX mail) and *user@dpss* (for UNIX mail to DPSS). These addresses must be in the subscriber directory.

In addition, the sendmail configuration file must be changed to identify the mailer for the DPSS and UNIX gateways.

2.4.2 DPSS/Mail and Sendmail

An optional feature allows all DPSS mail to be delivered through **sendmail**. This option can be configured on a system-wide basis by setting the **sm** option in **mail/$config/ver.17**.

2.4.3 DPSS/Mail and Alis

If you are using the Alis gateway you must install **sendmail,** since mail from DPSS to Alis is delivered via **sendmail.**

There are two types of configurations for mail that enters the Alis mail system: the user and the gateway configurations.

Users communicate with the gateway that allows communication with Alis, using **sendmail_post.** This configuration requires a line, similar to the following, in **sendmail.cf** to target the gateway node:

```
Malis,      P=/usr/lib/mailer/sendmail_post, F=lFDhum, S=10, R=20,
A=sendmail_post $u -n //alis_gateway_node
```

*Alis is a trademark of Applix, Inc.

In the gateway configuration, **sendmail_deliver** executes on the node running Alis, which serves as the gateway. The **sendmail_deliver** program waits for messages and then executes the local **sendmail**. The gateway executes the Alis import program.

The **sendmail.cf** on the gateway node for Alis mail looks similar to the one below:

```
Malis,      P=/usr/lib/mailer/alis_import, F=lFDhum, S=10, R=20,
A=alis_import $u
```

You must start **sendmail_deliver** via the appropriate start-up file on the gateway node. For example, the simplest way to start **sendmail_deliver**, is to include the following line in the **/etc/rc** file:

```
# cps /usr/lib/mailer/sendmail_deliver
```

2.5 Aegis Print Services

Aegis print services are handled by print managers (**prmgr**) and print servers (**prsrvr**). Print managers coordinate user print requests generated by the **prf** command and control one or more print servers. Print servers determine the print parameters and send print requests to a selected printer. Print servers are bound to the print managers in the printer configuration file.

When the print manager receives print requests, it checks to ensure that the requested printer exists. If it does not, the job is rejected. If the printer exists, the print manager notifies an appropriate print server. The print server processes the print job and informs the print manager of the ongoing status of the print job.

Both **prmgr** and **prsrvr** are supplied in the **/sys/hardcopy** directory.

Print managers and servers are each described separately in the following sections.

NOTE: For more information on Aegis print services, refer to the *Printing in the Domain Aegis Environment* manual.

2.6 The Print Manager: prmgr

Print managers are started with the **/sys/hardcopy/prmgr** command. As an option to the **prmgr** command, you supply the print manager configuration file name.

2.6.1 Print Manager HELP

Help for using the print manager is available online. For general information about print managers, enter

```
$ help prmgr
```

2.6.2 The Print Manager Configuration File

The configuration file defines the manager's spooling node and logical name and, if appropriate, invokes a print daemon that allows printing of pre-SR10 print jobs.

The print manager configuration file contains three items:

- The print manager logical name
- The print manager's spooling node, a node that includes a **/sys/print** directory (not just a link to that directory)
- An optional switch, pre10q, which allows the SR10 print environment to accept pre-SR10 print jobs

You should define a configuration file for each print manager. Their names should identify the type and location of the printers. A sample file defining a spooling node named **//flash** and a print manager named **r&d1** is shown in Figure 2-1.

```
spool_node = //flash
prmgr_name = r&d1
pre10q
```

Figure 2-1. Sample Print Manager Configuration File

The Print Manager Logical Name

You should assign print manager names that identify the type and location of the printers serviced by the print manager.

2.6.3 Starting the Print Manager

Like all processes, you can start the print manager process automatically at node startup, using the appropriate startup file (usually **/etc/rc**). A server process begun this way starts when the node comes online and continues after logout. This is the recommended start–up process.

You can also use other methods to start the print manager. To start the print manager from a process window, enter

```
$ /sys/hardcopy/prmgr -cfg configuration_filename
```

To start the print manager as a foreground process from the DM command line, enter

```
Command: cp /sys/hardcopy/prmgr -cfg configuration_filename -n process_name
```

To start the print manager as a background process from the DM command line enter

```
Command: cps /sys/hardcopy/prmgr -cfg configuration_filename -n process_name
```

In all the commands shown above, the variables have the following meanings:

configuration_filename	The path name of the print manager configuration file.
process_name	The name assigned to the print manager process. We recommend that you use the logical name specified in the configuration file.

2.6.4 Stopping the Print Manager

If the print manager is started in a manner that continues after node logout, it will continue running until either the node stops running (either intentionally or because of a system crash) or until it is intentionally stopped with the shell command **sigp**.

To use this command to stop a print manager, enter

```
$ sigp process_name -q
```

2.7 The Print Server: prsvr

The print server process, **prsvr** (**/sys/hardcopy/prsvr**), handles the processing of files submitted by the print manager for printing.

2.7.1 Print Server HELP

Help for using the print server is available online. For general information about print servers, enter

```
$ help prsvr
```

For details about printer configuration files, enter

```
$ help prsvr config
```

For a list of printers supported by the print server, enter

```
$ help printer
```

2.7.2 Starting the Print Server

Print servers are started with the **prsvr** command. An option to this command specifies the name of the printer configuration file that defines the printer and its print parameters.

Typically, print servers are started as background processes during node startup from the apropriate start-up file (usually **/etc/rc**). This is the recommended start-up method for printers that are available to the entire network at all times. There are other start-up methods you can use. Select these methods on the basis of the print server's host node and the attributes you want the print server process to have. Each start-up method is described below.

Starting the Print Server at Node Startup—The Recommended Method

You can start a server process automatically at node startup time by using **/etc/rc file.** A server process begun this way starts when the node comes online and continues after logout.

As supplied, the **/etc/rc.use** file contains the following line:

```
# cps /sys/hardcopy/prsvr config_file_name -n print_server
```

Edit this line to remove the # (comment marker) and include the name of the printer configuration file just in front of the **-n** option. (The configuration file can reside in any directory; however, it usually is convenient to place it in **/sys/hardcopy**.) The sample below shows a line calling the configuration file named **/sys/hardcopy/spinwriter**.

```
cps /sys/hardcopy/prsvr/sys/hardcopy/spinwriter -n spinwriter
```

Starting the Print Server from the DM Command Line

From the DM command line, on the local node, enter

```
Command: cps /sys/hardcopy/prsvr configuration_filename -n process_name
```

Replace *configuration_filename* with the name of the printer configuration file and *process_name* with the name of the print server process.

If you do not use the **-n** option, the print server process (*process_name*) is called **print_server.***printername* by default. (*printername* is replaced with the device name specified in the configuration file.)

The server process begins immediately and continues after logout.

Starting the Print Server from the Shell Command Line

To start a **prsvr** process from the shell command line at the time you want to use the printer, enter the **prsvr** command in the following format:

```
$  /sys/hardcopy/prsvr configuration_filename
```

The *configuration_filename* is the name of the printer configuration file.

This print server runs only for the life of the shell's process, and quits at logout time.

Starting the Print Server from a Remote Node

The shell command **crp** starts the print server on remote nodes running **spm**. Use the **crp** command in one of the following three ways:

```
$ crp '/sys/hardcopy/prsvr configuration_file' -n process_name -on node_spec
```

This version of the command creates a process on the remote node and prompts the user to log in. Processes started with this way run in the foreground and end at logout. (They have the same attributes as a process started with the **cp** command from the DM input window.)

```
$ crp '/sys/hardcopy/prsvr' configuration file -n process_name -on node node_spec -cpo
```

This version of the command runs the server process in the background (there is no window). Processes started this way end at log out. (They have the same attributes of processes started via the **cpo** command from the DM input window.)

```
$ crp '/sys/hardcopy/prsvr configuration_file' -n process_name -on node_spec -cps
```

This version starts the server process in the background. Processes started this way continue running after logout and can be stopped with the shell command **sigp**. (Processes started this way have the same attributes of processes started via the **cp** command from the DM input window.)

2.7.3 Stopping the Print Server

If the print server is started in a manner that continues after node logout (for example, all methods except the **crp** command with the **-cps** option or the DM command), it will continue running until either the node stops running (either intentionally or because of a system crash) or until it is intentionally stopped with the shell command **sigp**.

To use this command to stop a print server, enter

```
$ sigp process_name  -q
```

Where *process_name* is the name of the print server process.

2.7.4 Print Server Configuration Files

Print server configuration files define page format and printer characteristics. In addition, they identify the SIO lines (if applicable) to which the printer is attached and the name of the printer's print manager. The system is supplied with a sample printer configuration file. You must define a configuration file for each of the printers attached to your system. These files allow you to start the printers and to use the **prsvr** options. Give a different name to each print server configuration file when there are several printers attached to a node. If you do not give the print server configuration file a name, the default name is **printer_config.data** (located in the current working directory). Figure 2-2 shows two sample configuration files.

2.7.5 Printer Configuration File Options and Arguments

Refer to the *Aegis Command Reference* and the PRSVR CONFIG help file for details on print server configuration file options and arguments.

PRINTRONIX PRINTER

```
prmgr_site      r&d1
printer_name    p
device_driver   printronix
paper_width     13.0
paper_length    11.0
top_margin      0.5
bottom_margin   0.5
form_feeds      1
file_banners    first
logo            none
```

IMAGEN CX PRINTER

```
prmgr_site        mkt1
printer_name      cx
device_driver     imagen
io_device_name    /dev/sio1
speed             19200
paper_width       8.5
paper_length      11.0
point             12.0
file_banners      last
logo              none
```

Figure 2–2. Sample Print Server Configuration Files

2.8 Remote Process Creation: The Server Process Manager

The Server Process Manager, **spm** (**/sys/spm/spm**), allows you to create a process on a node from another, remote node. On a DSP, **spm** starts when the operating system is loaded, and so it runs whenever the DSP is online. The **spm** starts the **mbx_helper** program. Since they have no monitors or keyboards, DSPs would be unusable without both of these server processes. Once **spm** is started, you can create processes from a remote node by using the shell command **crp**, as well as logging in to the node for debugging purposes or to maintain servers.

2.8.1 Starting and Stopping spm

To start **spm** from the DM command line, enter the following:

```
Command: cps /sys/spm/spm
```

The **spm** begins immediately and continues after logout.

To invoke **spm** from the **/etc/rc** file, uncomment the following line in the file:

```
# cps /sys/spm/spm
```

The **spm** begins when the node is booted, and it continues under normal conditions until it is intentionally stopped with the shell command, **sigp**, as shown:

```
$ sigp server_process_manager -q
```

2.8.2 The shutspm Command

In addition to the **sigp** command, you can also use the **shutspm** command to shut down the Server Process Manager on a remote server node. Unlike **sigp**, which shuts down only the **spm** process, **shutspm** shuts down the **spm** and then performs an orderly shutdown of the node. When you reboot the node after a **shutspm** shutdown, it reboots without performing a **salvol**. (If you require a **salvol**, shutdown the node by pressing its RESET button.)

To shut down the **spm** with **shutspm**, create a remote process (via the **crp** command) on the target node and enter the **shutspm** command.

To prevent **spm** from responding to the **shutspm** command, add the following line to the **/etc/rc** file on the node or the DSP:

```
no_shutspm
```

The spmshut_ec File

The **shutspm** command performs the shtudown by advancing an eventcount file, **'node_data/spmshut_ec**, to the point that executes an orderly node shutdown.

The spm creates the **spmshut_ec** file in the **'node_data** directory. If the default ACL for files created in this directory is %.%.%, **spm** will apply the following protection to the **spmshut_ec** file:

Subject ID	**Access Rights**
%.sys_admin.%	pwrx
%.%.%	r

This ACL limits **shutspm** shutdown to **sys_admin** log-in accounts, but permits any account to delete the **spmshut_ec** file whenever **spm** is not using it. If the default ACL for **'node_data** has been changed, **spm** creates the eventcount file with that default ACL.

If the **spmshut_ec** file already exists when **spm** starts up, **spm** does not change its ACL. This ACL application procedure provides some control over who may shut down a remote server while still allowing you to administer your system the way you choose.

Chapter 3

Administering Nodes in the Network

Contents

Chapter 3
Administering Nodes in the Network

This chapter describes how to administer nodes in the network environment. It includes information about these topics:

- The directory structure of the network and individual nodes
- Providing network services to node users
- User log–in accounting
- Node activity when the node starts up (boot time) and when a user logs in, and how to use start–up and log–in files
- Troubleshooting node software behavior
- Servers used by nodes

3.1 The Network Directory Structure

The operating system expects information about nodes to be organized in a hierarchy called the **naming tree**. The two naming tree components, directories and files, combine to form pathnames. Directory names and file names must have fewer than 255 characters; pathnames must have fewer than 1023 characters. Figure 3–1 illustrates the network naming tree, including some of the standard software directories.

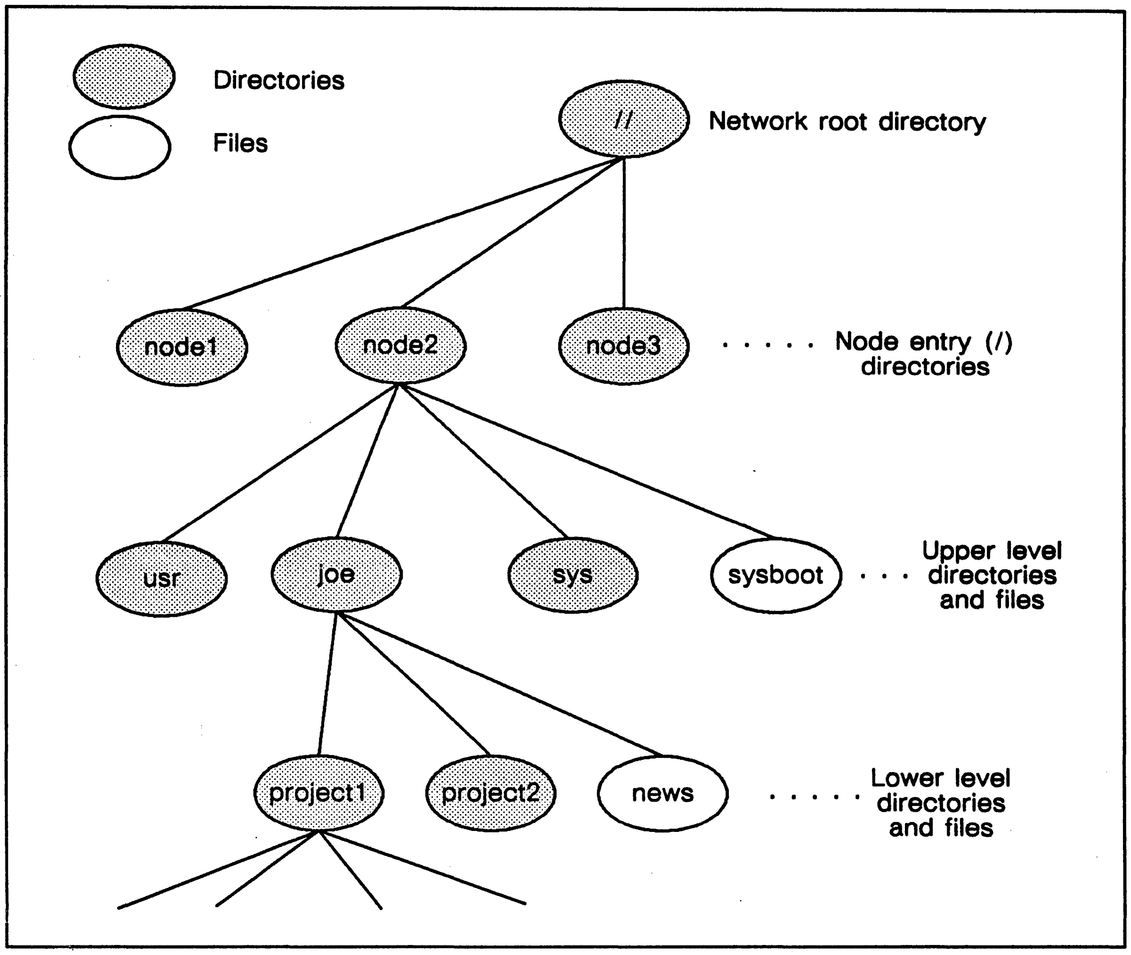

Directories
Files
//
Network root directory
node1
node2
node3
Node entry (/) directories
usr
joe
sys
sysboot
Upper level directories and files
project1
project2
news
Lower level directories and files

Figure 3-1. Network Directory Structure — the Naming Tree

3.2 Node Directory Structure

A node's directory structure comprises entry and root directories, upper level directories, and volume entry directories. This section discusses these directory types, distinguishes between physical and logical volumes, and explains two special characters in the file system: the ' (tick) in **'node_data** and the / (slash) identifier in the context of the directory structure.

3.2.1 Node Entry Directories and Root Directories

The node entry directory is the highest-level directory in a node's naming tree that can contain files, links, and other directories. The entry directory name is the same as the disked node's name.

The set of node entry directory names in your network is the network root directory, sometimes referred to as the top-level directory. Every disked node has its own copy of the root directory that the operating system uses to find objects stored on other nodes. As described in the previous chapter, you manage the names in the node root directory with the **ctnode** and **uctnode** commands, or if your network uses the **ns_helper** utility, with the **edns** tool.

3.2.2 Upper-Level Directories

Upper-level directories are one level below the node entry directory in the Aegis naming tree structure. Upper-level directories contain Aegis system software, system and programming libraries, and users' home directories. The system software installation procedures create the upper-level directories that the system needs to operate. In addition, you can use the **crd** command to establish upper-level directories on each node, such as home directories for different users.

You should restrict access to upper-level directories that contain system software by setting permissions on them so that they cannot be accidentally deleted or altered by users. Individual users may also want to protect all or part of their home directories by using permissions. Chapter 5 discusses protection.

3.2.3 Disk Volumes and Volume Entry Directories

Disked nodes have one or more **physical volumes**, that is, physical media. One disk drive is treated as a single physical volume; therefore, a node can have as many physical volumes as it has disk drive units. Each physical volume can be divided into **logical volumes,** independent partitions of the physical volume. You use the **invol** utility to create and maintain logical volumes on the physical media, and the **mtvol** and **dmtvol** commands to mount and unmount logical volumes.

Most nodes have one logical volume per physical volume, because this is usually the most efficient way to use disk space. However, you might partition a physical volume into logical volumes if you want to reserve part of a large disk for temporary files, or have another version of the operating system on the disk, or if you want to be able to run **salvol** on only part of the disk (so that **salvol** would run more quickly).

Each logical volume mounted on a node has a volume entry directory. This directory is the logical volume's highest-level directory. The first time you mount a logical volume, the **mtvol** command catalogs the name of the volume entry directory in the next higher directory in the naming structure. See Chapter 6 for further information about the **mtvol** and **dmtvol** commands.

3.2.4 The 'node_data and / Identifiers

Using Your Aegis Environment describes directories and links in detail. However, because the meaning of **'node_data** (tick-node_data) and / (slash) are especially important in the context of system administration, we go over the information here as well.

The meanings of the identifiers **'node_data** and / (alone or at the beginning of a pathname) are variable, and depend on the context in which you use them. Using them incorrectly, especially in links, can result in circular or otherwise incorrect references.

As the first character of a pathname, the slash character always refers to the node entry directory of the node where the process is executing. Similarly, '**node_data** always refers to the **/sys/node_data**[*.node_id*] directory for the node where the process is executing.

The / and '**node_data** conventions are powerful tools. The '**node_data** convention allows you to specify the **node_data** directory of the node you are working on, and that node only, even if the node is diskless. Similarly, the slash character refers to your working node's entry directory, independent of the current working directory.

The 'node_data Directory and Diskless Nodes

The actual pathname of the '**node_data** directory for a diskless node is distinguished from its disked partner's '**node_data** with a suffix that consists of its hexadecimal node ID. When you're working on a diskless node and specify '**node_data**, the operating system understands that to mean the **/sys/node_data.***node_id* directory on the disked partner.

Suppose there are three nodes: **brook**, **blue**, and **rainbow**, as shown in Figure 3-2. **Brook** is the disked partner node for the two diskless nodes **blue** and **rainbow**. A program or person working at node **brook** who reads the file '**node_data/startup.191** will read **//brook/sys/node_data/startup.191**. A program or person working at node **rainbow** (node ID e467) who reads the file '**node_data/startup.191** will read **//brook/sys/node_data.e467/startup.191**. A program or person working at node **blue** (node ID 632b) who reads the file '**node_data/startup.191** will read **//brook/sys/node_data.632b/startup.191**. (Remember, however, that a program or person working at either node **rainbow** or node **blue** who reads file **/sys/node_data/startup.191** will read that file on the **brook** node.)

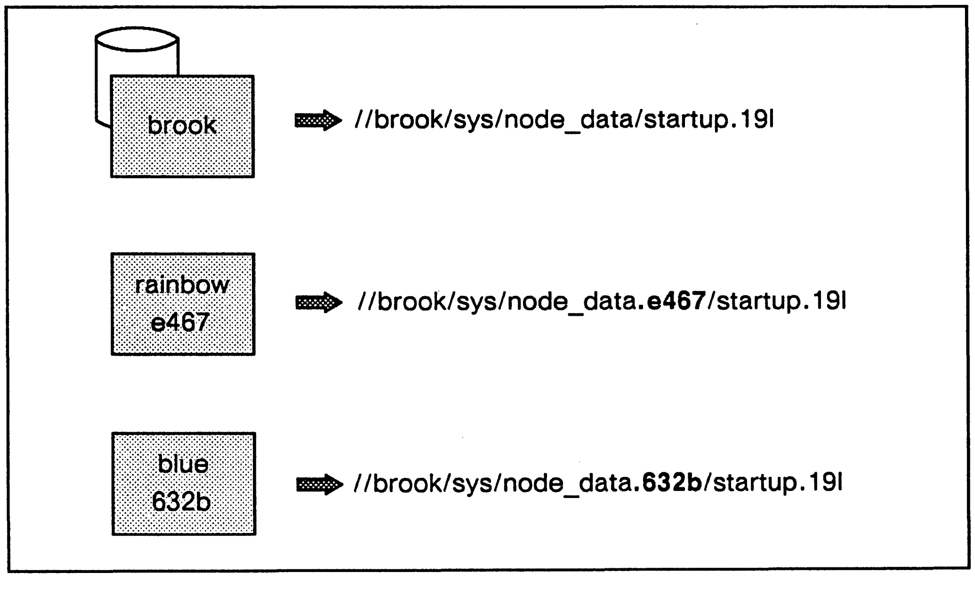

brook
//brook/sys/node_data/startup.191
rainbow
e467
//brook/sys/node_data.e467/startup.191
blue
632b
//brook/sys/node_data.632b/startup.191

Figure 3-2. Reading 'node_data from Diskless Nodes

The crp Utility

When you use the **crp** utility to execute commands or programs on another node, **'node_data** and the slash (/) refer to the **node_data** and entry directories of that remote node. For example, if you are working at the node **here_node** and use **crp** to create a remote process on the node **there_node**, the following command displays the directory listing for **//there_node/sys**, the remote node's /sys directory.

```
$ ld /sys
```

Circular and Unexpected References

Because **'node_data** and the slash have meanings dependent on the location of the process making the reference, it is possible to use pathnames that result in circular references. This is particularly true when you use links to either the slash or **'node_data** directory. For example, each node's **/tmp** directory is a link to **'node_data/tmp**. If you are working at node **bar**, you might try to use the following command to list node **foo**'s **/tmp** directory.

```
$ ld //foo/tmp
```

However, this command will actually list the contents of **//bar/sys/node_data/tmp** because **//foo/tmp** is a link to **'node_data/tmp**, and **'node_data** always refers to the **node_data** directory of the node executing the command, in this case node **bar**. Therefore, to list the contents of the **/tmp** directory of node **foo** when you are working at node **bar**, you must use the absolute pathname of the file in the command.

```
$ ld  //foo/sys/node_data/tmp
```

3.3 Node Software Structure

Figure 3–3 shows a typical node's directory structure, including system software and user entry directories, starting at the node entry directory level. This figure shows only the general way the software is organized; it does not include all software.

NOTE: All systems, including nodes that do not have a SysV environment installed, have a /sys5.3/bin directory. On systems that have only BSD or Aegis, this directory contains a minimum subset of System V commands that are required to install software. This enables solution suppliers to provide a single (SysV) script that will correctly install software on all nodes, independent of the installed environment

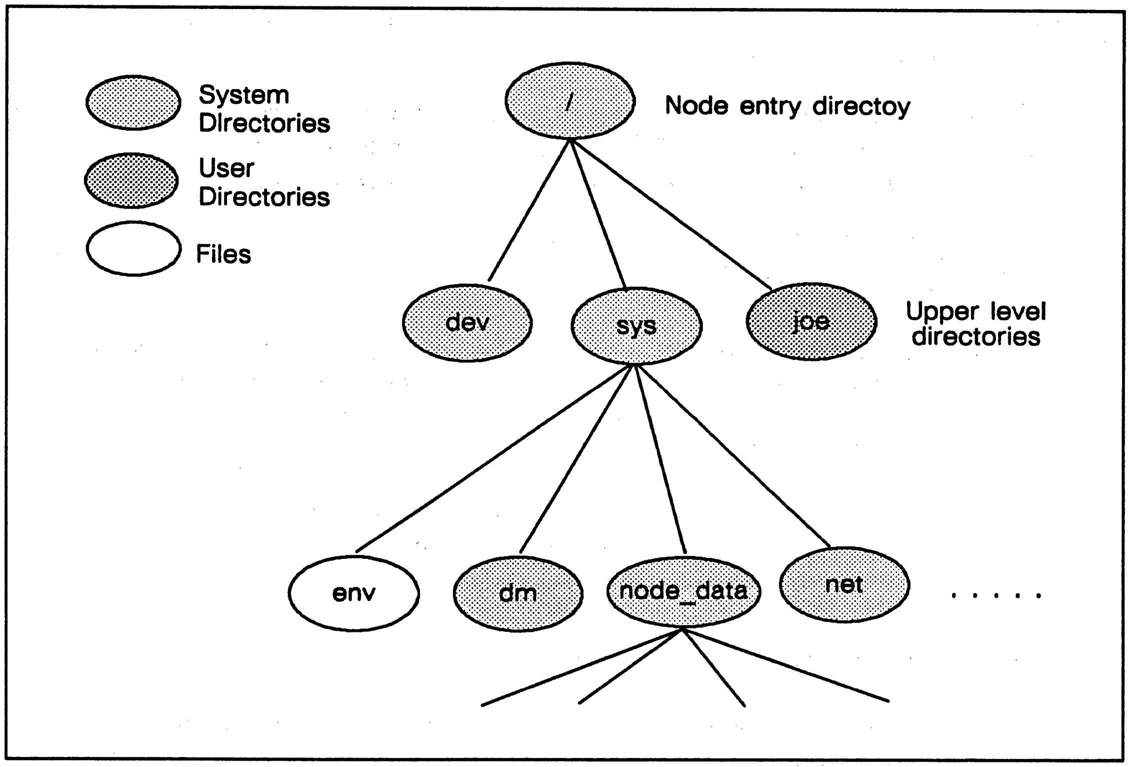


Figure 3-3. Disked Node Directory Structure

The exact structure of your directories and their contents will differ depending on the exact configuration of your Aegis system.

In the figure, the node entry directory contains the Aegis system software and other subdirectories that themselves contain Aegis system software. The /**sys** directory contains system software and many of the directories used for node and network management. Two subdirectories of the /**sys** directory, the Display Manager directory /**sys/dm** and the network management directory /**sys/net**, contain files and programs you use to create and administer network services. All installations will have the **'node_data** directory.

3.3.1 The 'node_data Directory

Every node, disked or diskless, has a **'node_data** directory. This directory contains files that the node needs to perform many system functions. These files include start-up and configuration files, per-node system files, interprocess communications mailboxes, and device files.

The actual pathname of a disked node's **'node_data** directory is

> *//node_name*/sys/**node_data**

The pathname of a diskless node's **'node_data** directory is

> *//partner_node*/sys/**node_data**.*node_id*

The optional *.node_id* part of the pathname enables a disked node to be a partner to one or more diskless nodes.

For example, if the disked node **golden** is the partner node for diskless nodes **sunapee** (node ID efd3) and **char** (node ID f2a4), **golden** would have the following three directories, each containing system operation information for its respective node.

/sys/node_data	(for golden)
/sys/node_data.efd3	(for sunapee)
/sys/node_data.f2a4	(for char)

NOTE: In later examples, when we refer to a node's **/sys/node_data** directory, we use the syntax **/sys/node_data[.***node_id***]** when we want to show that we are talking about the **/sys/node_data** directory for both disked and diskless nodes.

See Section 3.10, "Administering Diskless Nodes," for further information about managing the **/sys/node_data.[***node_id***]** directory for diskless nodes.

3.3.2 Temporary Files and Log Files in 'node_data

Apollo provides two directories for temporary files, **'node_data/systmp** and **'node_data/tmp**, and one for log files, **'node_data/system_logs**. The **'node_data/systmp** directory is used for temporary operational files created by Apollo software (system and other).

The **'node_data/tmp** directory (a link from **/tmp**) directory is used for temporary files created by the UNIX operating system. Files in both of these directories are deleted by the **/etc/rc** file when the system starts up.

The **'node_data/system_logs** is used by Apollo software to store log files.

Ensure that all users can write to the temporary files in all of these directories. Also, regularly purge the **'node_data/system_logs** directory of unnecessary files to keep the directory size reasonable.

By separating temporary directories from the other directories and files in **'node_data,** you can set protection for tempory files so that users can access them, but still maintain the overall tighter protection established for the **'node_data** directory.

3.4 Managing System Resources

You must manage system resources in order to distribute network resources such as databases, organize and protect user home directories, and organize and protect libararies of application software.

Frequently you manage system resources by creating and managing upper-level directories. The decisions you make about the locations of upper-level directories that contain network resources will affect the performance of your network. For example, some system libraries and databases are large or heavily used; you should position such resources strategically to maintain an even flow of network traffic and optimize disk space.

3.4.1 Using Upper-Level Directories

By assigning each library of application software its own upper-level directory, you can easily protect the software from accidental or unauthorized changes. You can set the permissions for the libraries so that only system administrators have full rights to change contents. For users who only run application programs, you can set their **home directories** to the proper application program library.

A **home directory** is an upper-level default working and naming directory set by the operating system when the user logs in. Users' upper-level directories do not become their home directories until you specify that fact in registry entries. Use the registry entries, as described in Chapter 4, to specify a user's upper-level directory as a home directory. Refer to *Using Your Aegis Environment* for a more detailed discussion of home directories.

3.4.2 Creating Upper-Level Directories

You create upper-level directories in the same way that you create other directories, by entering the **crd** command and specifying the directory pathname. For example, to create the upper-level directory **joe** on the node **//joes_node**, enter this command:

```
$ crd //joes_node/joe
```

3.5 Providing System Services

Server processes provide services to some or all of the nodes on a network. Servers normally run regardless of log-in and log-out activity. Server processes manage requests from **clients**, which may be programs or other processes. Clients request access to network resources such as data, peripheral devices, or communication pathways outside the network.

Server processes manage requests for data access and data transfer, gather network statistics, manage access to network resources such as printers, and manage communication paths outside the network.

3.5.1 Server Processes and DSPs

Server processes often run on Domain Server Processors (DSPs), server nodes that do not have displays and that are dedicated to running these processes. However, you can configure network server processes on any kind of node.

3.5.2 Numbers and Locations of Servers

The number of times you implement a server process depends on the particular requirements of your site. For example, an **ns_helper** process must run on each Domain network that is connected to make a Domain internet.

Decisions about the number of times to implement a server are closely related to the placement of server nodes within the network topology. For example, five printers can be managed from one, two, or five server nodes, depending on the locations of the printers. If your network covers a one–story building and a larger two–story building, you might want three server node locations, one in the small building and two in the larger building.

The location of servers within the network topology affects your ability to provide services to nodes and loops as they are switched in and out of the network. A node selected to run a network server process can be used for other activities.

It's not necessary, and it's usually not desirable, to place all network services on one or two nodes. In addition, server processes should run on nodes that are stable and secure.

3.5.3 Server Process Information

Section 3.13 presents server process reference information for system management servers. The reference section contains the following information for each server:

- A description of the server process
- Methods for starting, stopping, and reinitializing the server process
- Information about configuration files
- Information about options and arguments, and examples of their use
- Special considerations

Two servers, **ns_helper** and **netmain_srvr**, have interactive tools. The **edns** utility, which the system administrator's uses with **ns_helper**, is described in the *Aegis Command Reference*. The **netmain** tool, which is used with **netmain_srvr**, is described in Appendix A.

The reference information distinguishes between creating servers on nodes with displays and those without displays (DSPs) because the two types of nodes can require different start–up methods.

3.5.4 Methods of Starting Servers

Several methods are used for creating servers. The method used affects the server's attributes, whether it runs in the foreground or background, and whether it runs on a local or a remote node. (Table 3–1 summarizes the information on how to start servers.)

Starting Servers on a Local Node

You can start servers on a local node in the following ways:

- Insert the server's pathname in the **/etc/rc**, **/etc/rc.user**, or **/etc/rc.local** files. These files are executed as a part of start–up processing. This is the recommended way to start server processes. See Section 3.8, "Start–Up Procedures," for more information.

  Note that you can also start servers by inserting the appropriate DM create process commands in the **'node_data/startup**[.*type*] (for DM) and **'node_data/startup.spm** (for SPM) file. This method, however, is not recommended, since future versions of Aegis may not allow server startup in **'node_data** start–up files.

- Execute DM create process commands from the from the DM input window. The DM **cp** (create process in a window), **cpo** (create process only), and **cps** (create process server) commands start server processes on a user's node. Commands issued from the DM input window start server processes immediately.

- Execute the **/etc/server** command. If the DM is not available, you can use this command to start server processes. This command has the following syntax:

  /etc/server *server_name server_arguments* **&**

  If you use the **&** option, **/etc/server** starts servers with exactly the same attributes as servers started with the DM **cps** command. If you do not use **&**, it starts them as background processes. Note also that **/etc/server** has a **–p** option that allows you to start servers with the log–in Subject Identifier (SID) rather than **user.server.none.** Domain/OS identifies each account with an SID. The SID consists of a person name, group name, and organization name.

Refer to the *Aegis Command Reference* for complete information about the DM commands **cp**, **cpo**, **cps**, and the shell command **crp**. See the **/etc/server** manual page for complete information on the **/etc/server** command.

Starting Servers on a Remote Node

If the Server Process Manager (**spm**) is running on a node, you can create processes on that node from another location. (The **spm** runs on DSPs by default, so you can always create other servers or processes on a DSP from a remote node.)

To create processes from a remote node, use the shell command **crp** (create remote process). The **crp** command takes the **cp**, **cpo**, and **cps** local process commands as options to specify the attributes of the process created. For example, the command

```
$ crp –on //trout –me 'cps /etc/ncs/glbd'
```

starts the process named **glbd** (in the **/etc/ncs** directory) on the remote node named **trout**.

Refer to the *Domain Display Manager Command Reference* for complete information about the DM commands **cp**, **cpo**, **cps**, and the shell command **crp**.

Table 3-1. Server Process Start-Up Methods

Server Start-Up Method	**Process runs in foreground or background?**	**SID of Process (*id* = node ID)**	**Process runs on local or remote node?**
Paths to DM command files inserted in a start-up file:			
cpo	Background, runs whether or not anyone is logged in.	log-in account SID	Local
cps	Background, runs whether or not anyone is logged in.	log-in account SID	Local
DM commands, entered in the input window:			
cp	Foreground, ends on logout.	log-in account SID	Local
cpo	Background, ends on logout.	log-in account SID	Local
cps	Background, runs after logout (except for the **siologin** server).	**user.server.none.***id*	Local
Shell commands, if **spm** is running on the remote node:			
crp -cpo	Background, ends on logout.	log-in account SID	Remote
crp -cps	Background, runs after logout.	**user-server.none.***id*	Remote
etc/server command (can be used in the event the DM is not available)	Background, if the **&** option is used. Runs after logout (except for the **siologin** server). Foreground if the **&** option is not used.	**user.server.none.***i d* If the **-p** option is used, SID will be the log-in SID.	Local

3.5.5 Assigning Names to Server Processes

When you use the **-n** *server_name* option with the server creation commands you can provide a name for the server process. If you give a server process a name, you'll be able to easily identify it in the list displayed by the **pst** (process status) shell command. If you do not give the server a name, and it does not name itself by default, the operating system identifies it with a process number. We recommend that you use names for servers (either the defaults that some servers provide, or a name of your choice).

For example, the following **cps** command starts a **prsvr** process with the name **printer_server:**

```
$ cps  /etc/prsvr -n ' printer_name'
```

The **pst** command lists this process as "printer_server".

3.5.6 Using Shell Command-Line Features

While most server processes that start in the DM command line don't use shell command-line features, there are some that do. For such servers, the DM passes command-line standard options to the server program. See the individual server descriptions in Section 3.13 for information about servers that use command-line features from the DM command line.

3.5.7 Using Configuration Files

You can create a configuration file, or names file, for any server that takes options. The file executes when you start the server process. Some server processes require configuration files; these files are explained under the individual server descriptions in Section 3.13.

3.5.8 Maintaining Existing Servers

To check on the status of network server processes, use the shell command **pst**. The **pst** command lists all processes running on a node. The **-n** *node_spec* option for **pst** shows the processes running on a remote node. Use **pst** and its options if you suspect that a server is not running.

If the **pst** display does not show the server process, restart the server.

If **pst** does not list a server that you started, the server might not have started properly. ACLs that don't allow access to the server process SID or to the **'node_data** directory can prevent a server from starting. If you use the ACLs that we provide in the ACL templates, you should not experience this problem. However, if you change ACL entries and then a problem with server start-up occurs, check the ACLs assigned to the server program itself (for example, /sys/**alarm/alarm_server**) and the **'node_data** directory of the node on which you want to start the server. Any person (or process) starting the server program must have Execute rights to the server program, and Read and Write rights to the **'node_data** directory.

The **pst** command sometimes lists a server process as running even though the server has stopped. If this happens, stop the server process explicitly with the **sigp** (signal process) command, then restart the process from the DM command line. To stop a server process running on a remote node, use the **crp** command to log in to the remote node, and then use **sigp** to stop the process. Refer to the *Aegis Command Reference* for information about the **sigp** command.

3.6 Establishing a Node's Environment

During software installation, the system administrator selects the Domain/OS environment(s) (Aegis, BSD, or SysV) that will be available on the node. When Domain/OS software is installed on a node, you must determine the node's primary environment and system type.

3.6.1 The Node's Primary Environment

When multiple environments are installed on a node, one must be designated as the node's default primary environment. A value called "ENVIRONMENT" is used to specify the environment for all behavior except UNIX name resolution. ENVIRONMENT determines

- Default key definitions
- The user's default shell if no shell is specified in the user's registry entry
- Command search rules

The ENVIRONMENT value can be **aegis**, **bsd**, or **sys5**. (Note that ENVIRONMENT is not an environment variable but is maintained as a per–process value.) If only one environment is installed, that automatically determines the value of ENVIRONMENT. No choice is needed and changing the value is not meaningful.

3.6.2 The Node's SYSTYPE

SYSTYPE is an environment variable used to specify UNIX name resolution for many varient links that use $(*systype*). Valid values for the environment variable are **bsd4.3**, sys5.3, or blank.

3.6.3 The etc/environ File

ENVIRONMENT and SYSTYPE values are specified in **/etc/environ** (which is actually a link to **'node_data/etc/environ**). For example, the file might say:

```
ENVIRONMENT = aegis
SYSTYPE = sys5.3
```

Valid ENVIRONMENT and SYSTYPE values are governed by what is installed on the node and the selection of the primary environment at installation time. Table 3–2 shows the valid combinations of primary environment, addtional environments, ENVIRONMENT, and SYSTYPE.

Table 3-2. Relationships of ENVIRONMENT and SYSTYPE Values

Primary Environment	**Additional Environment(s)**	**Value of ENVIRONMENT**	**Value of SYSTYPE**
Aegis	None	**aegis**	blank
Aegis	BSD	**aegis**	**bsd4.3**
Aegis	SysV	**aegis**	**sys5.3**
Aegis	BSD and SysV	**aegis**	Choice of **bsd4.3** or **sys5.3**
BSD	NA	**bsd**	**bsd4.3**
SysV	NA	**sys5**	**sys5.3**

NOTE: If **ENVIRONMENT** is set to one of the UNIX values, **SYSTYPE** must correspond to that value.

If the **/etc/environ** file is not found, the defaults are ENVIRONMENT is **sys5** and **SYSTYPE** is **sys5.3**. The value of ENVIRONMENT can be displayed by using the **/etc/environment** command; thus shell specific scripts can query the current environment.

3.7 Establishing a User's Environment

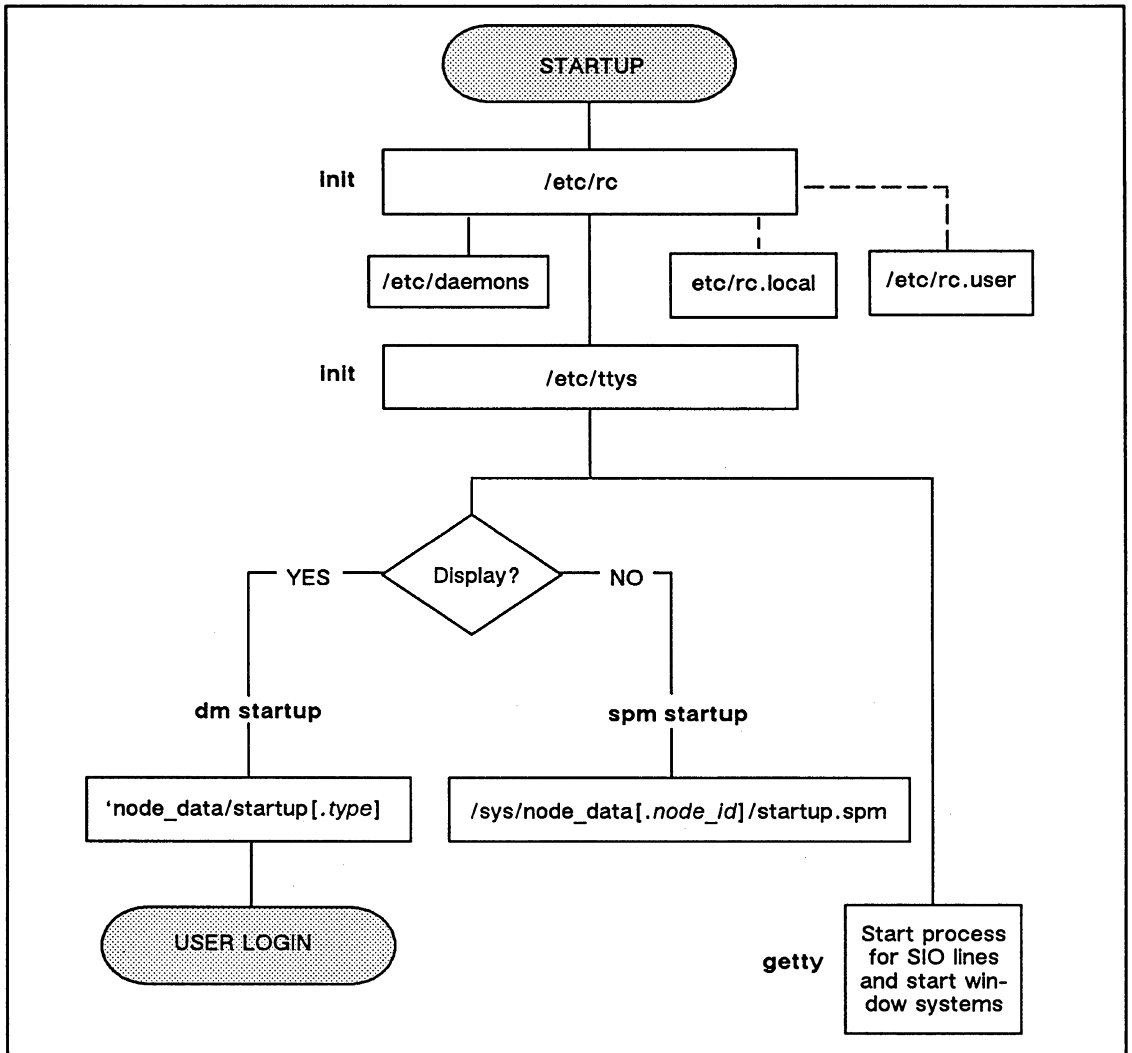
If a node has multiple environments installed, users can choose to change the primary environment in effect while they are logged in at that node. The file **$(HOME)/.environ**, like the **/etc/environ** file, can contain lines specifying ENVIRONMENT and SYSTYPE. If this file is found during user log-in processing, the environment and system type specified are used as the user's environment, as long as the specified environment is installed on the node. (If the environment is not installed, the node default environment is used.)

3.8 Start-Up Procedures

This section describes node start-up procedures, user log-in procedures, and related files.

3.8.1 Node Startup

The **/etc/init** program initiates node startup. This program first executes a Bourne shell script named **/etc/rc** (which in turn, will execute other scripts) and then executes **/etc/ttys** (which also executes other scripts). Figure 3-4 illustrates node start-up procedures. Each script file in the start-up process is described in the subsections following the figure.



Note: Solid lines (———) indicate automatic operations. Dashed lines (-----) indicate paths where the preceding file invokes the following file (for example, /etc/rc invokes /etc/rc.user).

Figure 3–4. Node Start–Up Files and Operations

3.8.2 The /etc/rc File

The **/etc/rc** file is a Bourne shell script executed as the first step in node startup by **/etc/init.** The **/etc/rc** file starts servers and other global processes as **root.wheel.none.** This file must also contain the commands that

- Delete any files in the **'node_data/tmp** and **'node_data/systmp** directories.
- Start any servers that require root access rights. (Root access rights enable a process to create, delete, or access any object on the system.)
- Invoke the **/etc/rc.local** and **/etc/rc.user** files.

Starting Servers in /etc/rc

The **/etc/rc** file contains lines that invoke servers. When the file processes these invocations, it first looks in the **/etc/daemons** directory to see if a file named for the server exists there. Since **/etc/rc** is owned by root and you must be root to edit it, **/etc/daemons** lets ordinary users (without root access) control which processes run from **/etc/rc** on their nodes.

The **/etc/daemons** directory contains special "stub" files named for the servers that should be started. Unless a file with the server's name is in this directory, the server will not be started, regardless of whether it's in the **/etc/rc** file. Node users can add or delete files to control the starting of server processes by **/etc/rc**.

3.8.3 The /etc/rc.local File

The **/etc/rc.local** file, invoked by **/etc/rc** during start-up processing, starts TCP/IP and its related servers. The commands in **/etc/rc.local** will execute with root access rights. Note that the **/etc/daemons** directory must contain a file named **tcpd** for the TCP/IP server to start. See *Using TCP/IP Network Applications* and *Configuring and Managing TCP/IP* for more information on TCP/IP.

3.8.4 The /etc/rc.user File

The **/etc/rc.user** file, invoked by **/etc/rc** during start-up processing, contains commands to start servers and other global processes that run as **user.server.none**, not as root. This file can be edited by users with other than root access, and does not need entries in **/etc/daemons** for the servers it starts.

3.8.5 The /etc/ttys File

The **/etc/ttys** file is executed by **init** during start-up processing. The **/etc/ttys** file starts the DM or SPM, **getty** processes for specified SIO lines, and possibly window systems.

Figure 3-5 shows the contents of a sample **/etc/ttys** file.

```
console   "/etc/dm_or_spm"      apollo    on       secure
tty01     "/etc/getty d1200"    dumb               off       secure
tty02     "/etc/getty std.9600" dumb               off       secure
tty03     "/etc/getty d1200     dumb               off       secure
```

Figure 3-5. Sample /etc/ttys File

The first entry starts the DM or the SPM, depending on whether the node has a display. The remaining lines in the file enable getty services on SIO lines. DM and SPM startup, SIO line initialization, and window system startup are described in the following subsections.

3.8.6 Display Manager Startup

When the DM manager starts, it automatically executes **'node_data/startup**[*.type*]. The *.type* option identifies the display type. Although you can use this file to start server processes, we recommend you use **/etc/rc.user** and **/etc/rc.local** instead. (If you do start processes in this file, the processes run regardless of log-in and log-out activity and run as **user.server.none**.)

Figure 3-6 shows the first lines in the start-up file we provide for the DN3000 monochrome node; the file is named **'node_data/startup.1280bw**. Corresponding start-up files for other types of displays draw the locations of the DM windows in different places; but, otherwise, the files are similar.

```
# STARTUP, /SYS/DM, default system startup command file for 1280x1024 monochrome
#
# Default is black characters on a white (or green) background.
  INV -ON
# Window positions for the DMs input and output windows.# Do not comment these out.
  (692,1011)dr;(1279,1023)cv /sys/dm/output
  (0,1011)dr;(639,1023)cv /sys/dm/input
  (640,1011)dr;(692,1023)cv /sys/dm/output;pb
```

Figure 3-6. Start-Up Script for DN3000 Monochrome (startup.1280bw)

3.8.7 Server Process Manager Startup

If you start the SPM on a node with a display, be aware that during SPM startup, the **'node_data/startup.spm** file will be executed. If this file contains the same commands as the DM start-up file, you will execute the same commands twice. If you regularly start the SPM from a node with a display, ensure that all the node start-up commands are in the **/etc/rc.user** file and delete the **startup.spm** file from the node.

If the SPM is running on a node (with or without a display), you can use the **crp** command from remote nodes to start processes on the node running the SPM.

Although it is not recommended, you can start server processes in the spm start-up file. The processes will run regardless of the node's log-in and log-out activity and will run as **user.server.none**.

Using ‘node_data/spm_control to Control Node Access

The **‘node_data/spm_control** can prevent unauthorized users from creating processes on or logging in to a node. If this file exists on the node running **spm**, all process creation and log-in requests are validated. Only users with a SID matching an entry in the file are allowed access; all others are rejected. If the file does not exist, all requests are allowed.

Each SID entry should be in the following format:

person.group.org

where a % character in a field matches anything.

Figure 3-7 shows a sample **‘node_data/spm_control** file.

```
# This file is used to control which users are allowed to start processes on
# a node running spm and spmlogin.
#
# If this file does not exist, no restrictions are in effect, and all users
# can gain access (ie pre SR10 behavior).
#
# If this file exists, then only those users whose sid matches a line in this
# file are allowed access. Sids are specified like ACLS, (user.group.org),
# one per line, with a % character matching anything. Note that all 3 fields
# must be specified.
#
# Entries are scanned from the start of the file, stopping on the first match.
#
# Thus the following line allows access to all users:
%.%.%
# Blank lines and lines beginning with # are ignored.#
```

Figure 3-7. Sample ‘node_data/spm_control File

SIO Line and Window System Startup

SIO lines connect “dumb” terminals to workstations, either directly or through a modem and telephone lines. The **/etc/ttys** file contains commands to start SIO lines and window systems. A sample line from the **/etc/ttys** file is shown below:

```
tty01     "/etc/getty d1200" dialup    on        secure
```

The line starts SIO port 1, specifying that it will be monitored by the **getty** program, that it will run 1200 baud, and that only users with root access will be allows to access the line.

The format for the command lines in the **/etc/ttys** file follows:

tty_line command terminal_type status [window=”*cmd_string*”]

tty_line	The terminal's entry in the device directory, **/dev.**
command	The command to execute for the line. This entry is usually **getty**. **Getty** is a program that performs such tasks as recognizing the baud rate, reading the log-in names, and calling **login**. For *command* you can also specify the start-up file for a window system terminal emulator or some other server process.
terminal_type	The type of terminal normally connected to that tty line, as found in the *termcap* database file.
status	A 2-entry option. For the first entry, specify **on** to enable the line; specify **off** to disable the line. For the second entry, to allow only root to log in on this line, specify **secure**. Do not quote this field.
window="*cmd_string***"**	The *cmd_string* is the pathname of the window system to start. If you specify a window system, it is started before any command is run for that line.

Tabs and/or spaces separate the fields. Use double quotes to enclose fields that contain more than one word (except the *status* field). If you omit a field, its value defaults to null.

NOTE: Previous releases used the SIO line servers **siomonit** and **siologin** to enable SIO lines. Although we recommend that you use the **/etc/ttys** file to enable SIO lines, you can still use **siomonit** and **siologin.** To do so, disable all the entries for serial lines in **/etc/ttys** by setting their *status* to **off**. Then, see Subsection 3.13.5 for information about using **siomonit** and **siologin.**

3.8.8 User Login

Figure 3–8 illustrates user log-in processing. The text following the figure describes each step in the process.

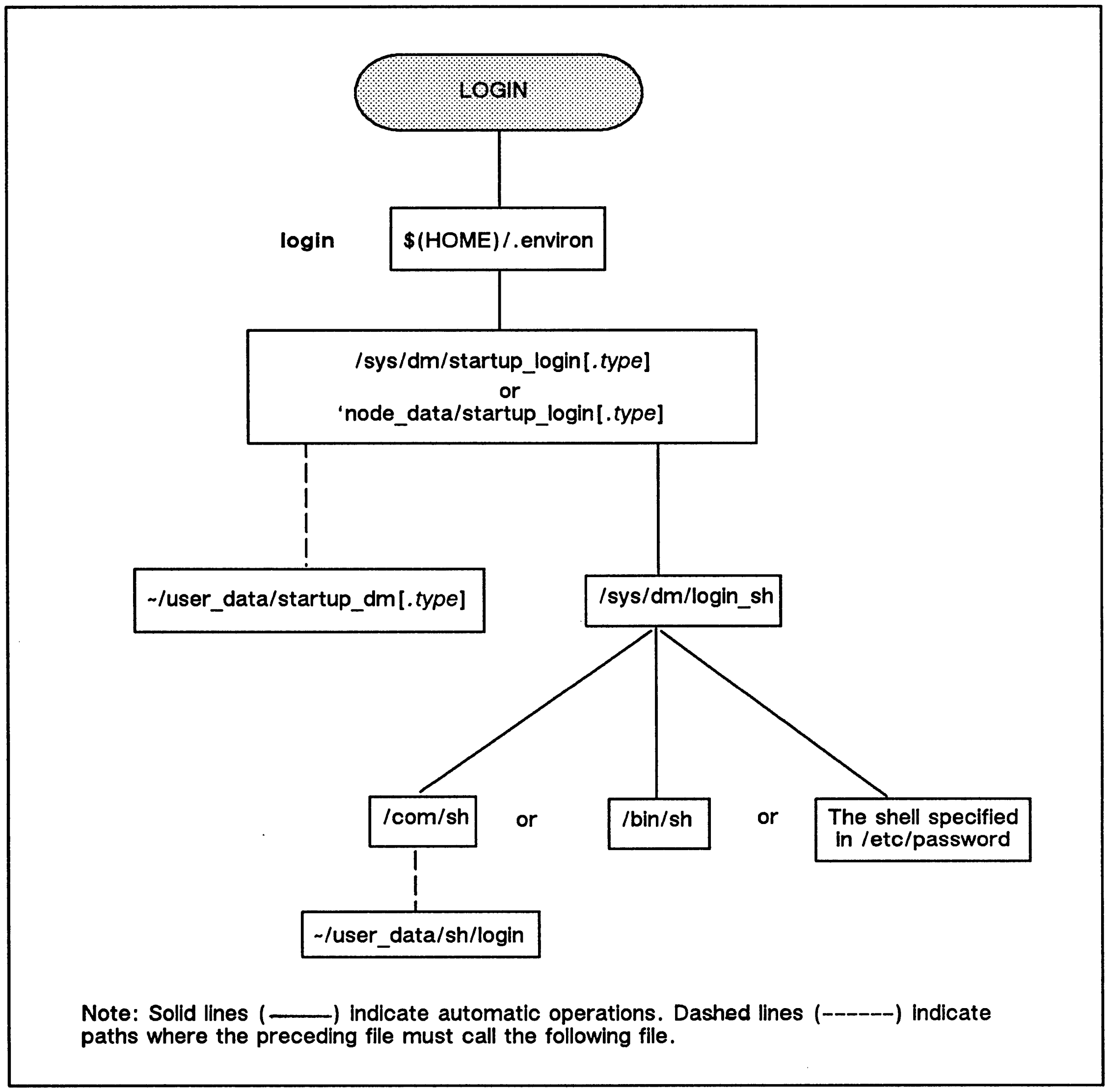

LOGIN
login
$(HOME)/.environ
/sys/dm/startup_login[.type]
or
'node_data/startup_login[.type]
~/user_data/startup_dm[.type]
/sys/dm/login_sh
/com/sh
or
/bin/sh
or
The shell specified in /etc/password
~/user_data/sh/login
Note: Solid lines (———) indicate automatic operations. Dashed lines (------) indicate paths where the preceding file must call the following file.

Figure 3–8. Node Start–Up Files and Operations

User Log–in Processing

1. When a user logs in to the node, the system

   a. Looks in the user's home directory for the **$(HOME)/.environ** file. If this file exists, it sets the environment and systype accordingly (if the corresponding environment was installed on the node).

   b. Looks for a shell definition in the user's registry entry. If one is not there, the appropriate shell for the current environment is used.

   c. Loads the key definitions based on the current environment.

2. The DM executes one of the following files:

   a. **'node_data/startup_login**[.*type*] — The DM first looks for this file. If it finds it, it executes the file.

   b. **/sys/dm/startup_login**[.*type*] — If the DM cannot find the file in **'node_data**, it uses this one.

   The **startup_login** file is used to perform tasks that need to be invoked every time someone logs in to this node. These tasks include specifying windows, creating shells, and running user-specific scripts. The **/sys/dm/startup_login** file is supplied with the system. You should copy it to each node's **'node_data** directory. Then you can modify it specifically for each individual node.

3. The file executed in Step 2 invokes a program named **/sys/dm/login_sh**. This program determines the user's environnment and log-in shell. To determine which log-in shell to execute for the user, **login_sh** looks at the user's entry in **/etc/passwd**. If the user's entry doesn't specify a default shell, **login_sh** executes **/com/sh** for the Aegis environment and **/bin/sh** for the BSD and SysV environments.

   As supplied, the *startup_login* file invokes **~/user_data/startup_dm**[.*type*]. This file, which contains commands private to the user to be executed at his login, is not supplied with the system, but must be created. To cause it to be executed, remove the pound sign (#) from the last line in the **/sys/dm/startup_login** file. (The line reads: **# cmdf user_data/startup_dm**[.*type*]). The SPM does not execute this script during remote logins. See *Using Your Aegis Environment* for more information about this start-up file.

For the Aegis shell only (**/com/sh**), users can create a file of shell commands (**~/user_data/sh/login**) to be executed upon login. For more information about this file, see *Using Your Aegis Environment*.

Key Definitions

When a user logs in to the node, the DM sets the default key definitions for the node according to the system type specified in **/etc/environ** or (if one exists) **$(HOME)/.environ**. Three sets of standard key definitions exist, one for each environment. When the user logs out, the DM resets the key definitions to the system type specified in **/etc/environ**.

3.8.9 Log-Out Script Processing

The DM processes log-out scripts. The log-out script must exist in the **'node_data** directory, and the script must be named **startup_logout**[.*type*], where .*type* is the appropriate display type suffix. You cannot start up new processes with the DM **cp, cps,** or **cpo** commands from this script.

3.9 Start–Up File Summary

Several types of start–up command files exist: files executed by the operating system at boot time, files executed by the DM and the SPM, and user files executed at log–in time, or when a shell is started.

Table 3–3 lists the command files that can be used at these times. Some of these files run automatically; others must be specified in start–up files. The following subsections describe the files that execute when the DM or SPM start running and when you log in to the DM. *Using Your Aegis Environment* describes the shell start–up files and their functions in detail.

Table 3–3. Start–Up Files

File	When Run	Comments
/etc/rc	System boot	Runs automatically
/etc/rc.local	System boot	Must be specified in **/etc/rc**
/etc/rc.user	System boot	Must be specified in **/etc/rc**
/sys/node_data[.node_id]/startup[*.type*]	DM startup	Runs automatically
/sys/node_data[*.node_id*]**/startup.spm**	DM/SPM startup	Runs automatically on DSPs
/sys/dm/startup_login[*.type*]	User login	Runs automatically
/sys/node_data/startup_login[*.type*]	User login	Runs automatically
/etc/login/sh	User login	Runs automatically
~/user_data/startup_dm[*.type*]	User login	Must be specified in **startup_login**
~/user_data/sh/login	User login	Runs automatically
~/user_data/sh/startup	By /com/sh	Runs automatically

NOTE: A *.type* argument in a pathname represents a node display type identifier; a *.node_id* suffix to **/sys/node_data** represents the hexadecimal identifier of the diskless node for which the directory holds information.

3.9.1 Node Display Types and Their Start–Up Files

Because display formats differ, particularly in their pixel dimensions, different nodes require different information in their start–up files. For example, the width of the DM windows vary among display types. For this reason, each node and DM login start–up file has several different versions, one for each display type. The display type is indicated by a suffix added to the start–up file name. Therefore, **/sys/node_data/startup.191** is the DM start-up file for nodes with a 1024-pixel by 800-pixel landscape display. Table 3-4 lists the display-type suffixes and indicates the node models to which they apply.

Table 3-4. Start-Up File Suffixes

File Suffix	**Node Types**
.spm	Server nodes
.1280bw	Monochrome DN3000 and DN4000
.1280color	DN580
.191	DN300, DN320, DN330, DN460, DN550, DN560, DN570, Color Domain 3000 and Domain 4000
.color	DN660

The installation procedures copy all versions of each start-up file onto a node. This procedure ensures that any node can be a partner for any type of diskless node. Similarly, it is useful to have multiple versions of the DM log-in files if you are likely to log in on nodes with different display types.

3.9.2 Templates for Start-Up Files

The installation procedures automatically create several start-up template files. Table 3-5 lists the files.

Table 3-5. Start-Up File Templates

File	**Comments**
/sys/dm/startup_templates/startup[.*type*]	Copied to **/sys/node_data.***node_id* for diskless nodes
/sys/spm/startup_templates/startup.spm	Copied to **/sys/node_data.***node_id* for diskless nodes
/etc/templates/rc	Copied to **/sys/node_data/etc** for diskless nodes
/etc/templates/inetd.conf	Copied to **/sys/node_data/etc** for diskless nodes

The files in the **/sys/dm/startup_templates** and **/sys/spm/startup_templates** directories serve two purposes:

- They are master copies of the DM and SPM start-up files. You should edit these master copies only if you are making changes that should propagate through to all nodes in the network and to all nodes to be added in the future. If you are changing the start-up files for individual nodes, edit the copy created for the specific node, not the master copy.
- They are used as the source for the files copied by **netman** whenever it creates a **/sys/node_data.***node_id* directory for a diskless node. For more details on this see Section 3.10, "Administering Diskless Nodes."

3.9.3 Start-Up File Format

The default versions of the start-up files that are created when you install the Aegis software contain most of the commands that you are likely to need. However, most of these commands are commented out in the files with a pound sign (#). You must delete the pound sign from the start of the line to allow the command to execute.

NOTES: The DM, the Server Process Manager (**spm**), and the shell are all case sensitive. Use lowercase characters when referring to system commands and files.

Changes that you make in a start-up file do not take effect until the next time the file is run. For example, changes to the DM start-up files do not take effect until the next system boot. To start a process before the next operating system boot, use the **DM cp**, **cps**, or **cpo** command.

3.10 Administering Diskless Nodes

There is no apparent difference between working on a diskless node and a disked node. However, before you can use a diskless node, you must configure the node and start the processes that support the diskless node's operation. The following subsections describe the rules and techniques for managing diskless nodes and their partners. The last subsection describes a procedure for configuring a diskless node's partner.

3.10.1 Diskless Node Operation

When a diskless node displays the log-in prompt, all the programs required for its operation are in place. While *Using Your Aegis Environment* gives a complete description of diskless node bootstrap operation, the following summary indicates what happens after you power on a diskless node in NORMAL mode.

1. The diskless node's Mnemonic Debugger broadcasts a message requesting a volunteer disked node partner.

2. Each disked node running the **netman** program listens for such a "request for volunteer" broadcast. The disked node answers the request if the requester's hexadecimal node ID is in the disked node's **/sys/net/diskless_list** file.

3. The diskless node loads **netboot**, its version of the operating system boot program from the partner, and proceeds with the bootstrap operation.

4. If the *//partner*/sys/**node_data**.*diskless_node_id* directory does not exist (for example, if the diskless node has never booted from this partner), the **netman** program creates the directory and copies the node **startup**[.*type*] file from the **/sys/dm/startup_templates** directory.

5. The diskless node's DM executes the commands in the *//partner*/sys/**node_data**.*diskless_node_id*/**startup**[.*type*] file.

6. The diskless node runs in the same manner as a disked node, using the partner node's disk for its system software.

3.10.2 Establishing Diskless Nodes and Their Partners

A diskless node's partner node provides the system software and disk services for the diskless node. The partner does not necessarily store any of the diskless node user's files. Each partner node must run the diskless node server, **netman**. Partners can be nodes with, or without displays. (Nodes without displays are DSPs.)

A partner node must have the correct system software for the diskless node type. For example, if the diskless node is a DN570 and the partner node is a DSP90, the partner node must have both a **/sau3** directory and a **/sau5** directory. Similarly, the partner's /sys directory must have any microcode files required by the diskless node.

Each diskless node has its own **/sys/node_data**.*diskless_node_id* directory on the partner node. The **'node_data** directory on each diskless node resolves to **/sys/node_data**.*diskless_node_id*. If you change diskless node partner assignments, delete the **/sys/node_data**.*diskless_node_id* directory from the original partner's **/sys** directory.

The home directories of diskless node users can be located on any node in the network, and do not have to be on the diskless node's partner node. Whenever possible, locate the home directories on the same loop as the diskless node. If a diskless node has one or more regular users, their personal log-in start-up scripts, **user_data/startup_dm**[.*type*], should be in their home directories.

Specifying Partners

A disked node can be a partner to one or more diskless nodes. You control the assignment of diskless nodes to partners through the disked node's **/sys/net/diskless_list** file, such as the one shown in Figure 3-9. The disked node will volunteer to be a partner for any diskless node whose node ID is in the disked node's **diskless_list**.

```
# This is the diskless list for the network server on node 6b2d.
#
# The first token in each line of this file is examined by NETMAN
# when it receives a network bootstrap volunteer request. If the
# token is a valid node ID, then NETMAN will volunteer to help
# that node when it calls. Lines that do not begin with a valid
# node ID will be ignored. The use of the comment line
# character "#" is recommended.
#
# The nodes that this file authorizes NETMAN to volunteer help for
# are:

3f4
eff21
4d76
```

Figure 3–9. Sample /sys/net/diskless_list File

Choose the partners for diskless nodes carefully. For example, a partner should be in the same network loop as the diskless node; then, if you switch the loop out of the rest of the network, the diskless node can still function.

> NOTE: If you put a diskless node's node ID on more than one diskless list, you cannot predict which node will become the partner when the diskless node boots. As a result, the diskless node's '**node_data** directory and the contents of that directory, could change whenever the node reboots. Therefore, you must configure the diskless node correctly on each possible partner node.

The /sys/node_data.node_id Directory on New Partners

If a diskless node's partner does not have a **/sys/node_data.***node_id* directory when the diskless node boots using the partner, **netman** automatically creates one. If this directory does not contain the minimal set of files required by the diskless node to operate, **netman** creates them. **Netman** also copies the **startup**[.*type*] file from the partner's **/sys/dm/startup_templates** or **/sys/spm/startup_templates** directory. See Subsection 3.13.3 for information on **netman**.

Providing a New Partner for a Diskless Node

Use Procedure 3–1 to configure a partner for a new node, change a partner for an existing node, or to provide an additional partner for an existing node. If a service representative installs a diskless node, he or she creates a partner for the node; you might want to use a different partner.

Procedure 3–1. *Providing a Permanent Partner for a Diskless Node*

Task 1: Determine the diskless node's ID

If the node is new, the node identification slip lists the node ID.

If the diskless node currently has a partner, you can determine the ID by entering the following command at the diskless node:

```
$ netstat
The node ID of this node is eff21.
```

Task 2: Log in to the partner node

If the partner has a display, log in at the partner node.

If the partner does not have a display, you can create a remote shell on the partner by using this command:

$ **crp -on** *//partner* **-me**

Task 3: Add the diskless node to the partner's list of authorized nodes

Edit the partner node's **/sys/net/diskless_list** file, and add the diskless node's node ID to the list.

If you are changing partner nodes, delete the diskless node's ID from the old partner node's **/sys/net/diskless_list** file. Also delete the old partner's **/sys/node_data.***diskless_node_id* directory.

Task 4: Start netman on the partner node

If the partner node is running **netman** already, go on to Task 5.

If the partner node is not running **netman**, do the following:

1. Remove the pound sign (#) from the following line in the partner's **/sys/node_data/startup***[.type]* or **/etc/rc** file:

   ```
   # cps /sys/net/netman -n netman
   ```

   The **netman** server will now start automatically whenever the partner reboots.

2. Start the partner node's **netman** server. (By doing this, you do not have to reboot the partner.)

   If you are at the partner node, enter the following command at the DM command prompt:

   ```
   Command: cps /sys/net/netman
   ```

   To start **netman** from a remote node, enter the following command. (You must use this command even if you used the **crp** command in Task 2.)

   $ **crp -on** *node* **-cps /sys/net/netman -n netman**

Task 5: Limit the size of the partner's memory pool (optional)

You can limit how many memory pages are left available for diskless node paging requests using the **netsvc** command. (Be aware, however, that this command affects the memory pool size for all situations in which files are used remotely on your node.)

To limit the size of the partner's memory pool, type

$ **netsvc -p** [*pool_size*]

where *pool_size* is the maximum number of memory pages that will be available for diskless node paging.

If you don't use this command, all of the partner's memory is available for paging requests from the diskless node.

Task 6: Name a diskless node (optional)

If you are installing a new diskless node, you can give it a name. Skip this step if the node already has a name.

If you do not use **ns_helper** on your network, enter the following command:

$ **ctnode** *node_name node_id*

If you do use **ns_helper** at your site, enter the following:

$ **ctnode** *node_name node_id* **-root**

After you finish this procedure, catalog the node on the network. Refer to Chapter 2 for details.

Task 7: Create node start-up and configuration files

NOTE: If you are changing the partner of an existing diskless node, **netman** copies these configuration files from the old partner to the new partner.

Edit the /sys/**node_data.***node_id*/**startup**[.*type*] file to configure the diskless node.

Task 8: Log off the partner node

Task 9: Reboot the diskless node

You can now start or restart the diskless node.

If you are changing an existing diskless node's partner, enter the following DM command to log off and shut down the operating system.

`Command:` **shut**

The Mnemonic debugger prompt (>) appears on the screen. Enter the following reset command:

> **re**

Press <RETURN> twice.

Restart the operating system by entering the following command:

```
> ex domain_os
```

The node now restarts, using the new partner node; you can now log in.

3.10.3 Managing Diskless Nodes and Partners

You should evaluate your diskless node partner assignments from time to time. Distribute assignments in a way that does not degrade the performance of either partner. The network maintenance server (**netmain_srvr**), and the **netmain** interactive tool, along with the **netsvc** shell command, can help you to manage partner assignments.

Diskless Node Management Commands

The **netmain_srvr** collects information about the number of diskless nodes assigned to any partner. The **netmain** interactive tool formats the performance data so that you can identify nodes providing more than their share of resources to diskless nodes in the network. For a detailed description of **netmain_srvr** and **netmain**, see Appendix A.

Whenever a diskless node cannot communicate with its partner, it displays a message to that effect. You will see this message if the partner stops running the operating system because of an intentional shutdown or system crash. Problems with the network can also cause the diskless node to display the message. Once the partner node is rebooted or the network is back up, use CTRL/F to refresh the screen on the diskless node and eliminate the messages.

> **NOTE:** You must reboot the diskless node whenever its partner node reboots.

Warning of a Partner Shutdown

It's good practice for the administrator of a partner node to notify diskless node users of intended shutdowns. Use the **send_alarm** or **/etc/wall** command to notify users of shutdowns. For example, if your are shutting down your disked node, the following command will warn all diskless nodes that use your node as a partner:

```
$ send_alarm 'Partner node disked is shutting down in 2 min. Please log out.' -mydi
```

Requesting a Specific Partner

If for some reason a diskless node's regular partner is not available, the diskless node can request as its partner another disked node that runs **netman**, even if that node does not have the diskless node on its partner list. You can also use this procedure to boot a disked node as a diskless node, which you might need to do if, for example, the node's system software is corrupted, or you are initializing the disk across the network. This is a temporary measure, however; you should not use this procedure in place of the **diskless_list** file.

Use the following steps if you must boot a diskless node by requesting a specific partner:

1. Boot the diskless node in SERVICE mode or, if the node is running the DM, use the DM **shut** command to enter the Mnemonic Debugger.

2. Enter the following command when the Mnemonic Debugger prompt (>) appears:

   > **di n** *node_id*

   where *node_id* is the hexadecimal ID of the partner node you are requesting.

3. Enter the following command to restart the node:

   > **ex domain_os**

3.11 Node Troubleshooting

If a node doesn't appear to be operating correctly in the network, you can diagnose the problem in a number of ways. This section does not attempt to be exhaustive, but it does outline some common faults that occur and how to correct them.

Check Connections and Power

Check the brightness control on the monitor. Make certain that cables are attached correctly and securely. Make sure the power is on.

Check LEDs

If the node has LEDs (such as the DN3000 series), see if they're active. If they are, suspect problems outside the node, perhaps with the network.

Try to Communicate with the Hung Node

Try to list the entry directory of the hung node. For example, if the node name is **chinook**, try the following command line from another node on the same network:

```
$ ld //chinook
```

If you receive the message "name not found," try recataloging the node name, as shown in the example below, and then retry the listing. If you receive an "object not found" message, make sure you're not on a loop that's been switched out of the network.

```
$ ctnode hex_id chinook -r -l
```

Is the hung node running diskless? If so, the problem might be that the partner node has crashed, or that temporary network problems have disturbed the communication between the partner and the diskless node. Remember, too, that a diskless node can have a name, but doesn't have an entry directory; so, if you **ld** a diskless node name as if it were an entry directory (for example, **//diskless**), you will receive the message "object not found."

Check Processes

You can check the processes running on the node in one of two ways. Either use the **crp** command to log in to the node and run the **pst** or **dspst** process status commands, or use the **-n** *node_spec* option with either the **pst** or **dspst** command to look at the processes. Look for exceptionally heavy network traffic or for a process taking up an inordinate amount of CPU time.

3.12 Log-In Monitoring

Aegis provides node log-in moniotoring for log-in attempts via

- The Display Manager
- A window
- The **spm**
- An SIO line

When you configure log-in monitoring, you can specify whether records of successful and unsuccessful logins or only unsuccessful logins should be kept. You can also specify the sources of the logins you want to monitor.

3.12.1 Configuring the Log-In Facility

The file **'node_data/login_log.config** specifies which attempts to log in should be recorded and the log file in which to store this information. The default log file is **'node_data/login_log**.

The **'node_data/login_log.config** file contains a line for each type of login that can be monitored. The sample line, shown below, is for logins via the Display Manager:

```
# -display [-inv_only]
```

To enable log-in monitoring, uncomment the appropriate lines. If you don't uncomment any lines and if you don't create the log file, no logging will take place. To record only invalid log-in attempts, remove the square brackets from around **-inv_only**.

To specify a log file other than **'node_data/login_log**, uncomment the following line and replace *path_name* with the name of the log file you want.

```
# -file path_name
```

Note that if the log file you specify does not exist, no logging will take place. You must creat an empty log file at the location you specify.

3.12.2 The Log-In Facility Log File

Information about the logins is stored in a log file. The default log file is named **'node_data/login_log**. Note that the log file grows without bounds and must be managed.

For each login, the log file contains

- The date and time of the login
- An indication of whether the login was from the Display Manager, a window, the SPM, or an SIO line using **siologin**
- The user ID under which the login was initiated
- An indication of whether or not the login was successful (unless you specify that only unsuccessful logins should be recorded)

Figure 3-10 is a sample log file.

```
Friday, October 30, 1987  16:12:45 window  odonnell.osdev.r_d  invalid login attempt
Friday, October 30, 1987  9:30:09 window  jjdjd  invalid login attempt
Friday, October 30, 1987  13:13:33 display  doug.osdev.r_d.5FE  logged in
Friday October 30,  17:36:26 window  root.staff_sr9.none.5FE  logged in
Friday, October 30,  1987  17:42:39 display  doug.osdev.r_d.5FE  logged in
Monday, November 2, 1987  15:09:28 sio  odonnell.osdev.r_d  logged in  connected to device:  /dev/sio1
Monday, November 2, 15:09:50 sio  odonnell.osdev.r_d.5FE  logged out from device:  /dev/sio1
Monday, November 2, 15:15:50 sio  intruder  invalid login attempt   on device: /dev/sio2
```

Figure 3-10. Sample Log-in Monitoring Log File

3.12.3 Log File Protection

To protect the log file from unauthorized access, give it fairly restricted protection. If you do this, you must also stamp the file as a log-in protected subsystem data object to allow the log-in program to open it and make entries. Also protect the **login_log.config** file appropriately. The following protection settings are suggested for the log file:

Acl for login_log:	**subsystem log-in object**
Required entries:	
`root.%.%`	`pwrx--`
`%.sys_admin.%`	`pwrx--`
`%.%.none`	`[ignored]`
`%.%.%`	`--r-k-`

(Note that, in the table, the **k** in the entry for **%.%.%** makes it unnecessary to protect the '**node_data** directory.) In some circumstances, you may want to add an entry that gives the node owner full rights to the file or make the node owner the owner of the file. See Chapter 5 for more information about protecting your files.

3.13 Server Reference Information

This section describes the following servers:

- Alarm Server — **alarm_server**
- Mailbox Server — **mbx_helper**
- Diskless Node Server — **netman**
- Tablet Server — **spb1**
- Serial I/O Line Servers — **siologin** and **siomonit**

You can obtain online help for any server by typing the following line, substituting the name of the server for *server_name*.

$ **help** *server_name*

Appendix A provides reference information on **netmain** and **netmain_svr**.

3.13.1 The Alarm Server: alarm_server

The Alarm Server (/sys/**alarm/alarm_server**) alerts you by popping a small alarm window on the screen and sounding an alarm.

Run the Alarm Server as a background process by starting it with the DM command **cpo**. Usually, you'll start the Alarm Server from an entry in a start-up file.

The Alarm Server can report on potential disk overflow, network problems, **netmain_srvr** observations, and brief messages from other users. Each condition is checked every four minutes by default, or at an interval that you set with the **-period** option.

By default, an alarm is accompanied by a distinctive tone pattern. You can disable the audible alarm or have all alarms alert you with a single short beep.

Alarm messages appear in windows of default sizes, accompanied by standard sound patterns. The first alarm window appears near the top left-hand corner of the screen. Use command options to alter the default window positions. A "pad closed" message appears at the bottom of the alarm window when the alarm is complete. Note that if you inadvertently press CTRL/Q in the window before the message appears, you will kill the Alarm Server.

Certain optional software packages also use the Alarm Server. See the release documents and manuals for any optional software you have purchased for details about the Alarm Server's operation with those products.

Starting the Alarm Server

To start the Alarm Server, enter the following from the DM command line:

```
Command: cpo /sys/alarm/alarm_server -[options]
```

The server process begins immediately and ends at logout.

To start the Alarm Server from a DM start-up file, include this command line in the file. You'll have to log out and log in again for the server to start.

The Alarm Server names itself "alarm_server" by default, so you don't need to specify the **-n** option with the **cpo** command.

Configuration Files

To execute Alarm Server options from a configuration file, create a file using the format illustrated in Figure 3-11.

```
-disk 75
-hw
-nm_srvr //netmain_srvr_node
-bell1
```

Figure 3-11. Sample Alarm Server Configuration File

Edit **~/user_data/startup_dm**[.*type*] to specify the configuration file. For example, in the user's home directory start–up file, the command

```
Command: cpo /sys/alarm/alarm_server '*~/user_data/options' -n alarms
```

uses the standard command line option, the asterisk (*), to point to the configuration file. The command names the process "alarms."

Alarm Server Options and Arguments

To receive alarms about any of the standard Alarm Server events, specify the event with one or more options described below. The default event reports are indicated by (D).

–disk[_full] [*nn*]

Posts an alarm when the disk containing the node's "/" directory is more than *nn* percent full. If you omit *nn*, **alarm_server** uses a default value of 95 percent full (5 percent free space). If you do not delete anything from the disk, or if the full–disk condition recurs, **alarm_server** notifies you again. After two notifications of a full–disk condition, **alarm_server** does not notify you again for at least one hour, even if the condition persists.

Default if omitted: Do not post disk–space alarms.

–hw[_fail]

Posts an alarm when some node on the network detects network hardware problems (as seen in the "last ring hardware failure" report from the **netstat –l** command). The **netstat** output lists the last hardware failure report detected on the network, whether it is new or not. The Alarm Server posts alarms only when there is a new report, indicating a current problem.

Default if omitted: Do not post a network problem alarm when there is a new hardware failure report.

–netmain [*pathname* ...]

Enables alarms from **netmain_srvr** observers. The *pathname*(s), if specified, represent text files containing lists of nodes that run **netmain_srvr**. If you don't specify a file for *pathname*, the Alarm Server uses the file **~/user_data/alarm_server.netmain_srvr_list**.

The files should contain node names or hexadecimal node ID numbers on different lines or separated by spaces on the same line. Comments in these files start with a left brace ({) or a pound sign (#) and run to the end of the line.

The Alarm Server reads these files only when it starts up. If you add or delete node names in these files after the server is running, changes do not take affect until the next time the server starts up.

Default if omitted: Do not enable alarms from **netmain_srvr** observers on node lists.

–nm_srvr *node_spec* [...]	Enables alarms from **netmain_srvr** observers on the node(s) specified with *node_spec*, which is a hexadecimal node ID or node name. Default if omitted: Do not enable alarms from **netmain_srvr** observers specified in the command line.
–nonetmain (D)	Prevents the Alarm Server from checking for observer alarms from the **netmain_srvr** program.
–nets	Notifies you when a network disappears from your internet or appears in your internet. This option is useful only in Domain internet environments.
–nonets (D)	Suppresses alarms describing changes in the internet topology.
–msg (D)	Allows the Alarm Server to receive messages from the **send_alarm** program.
–nomsg	Prevents the Alarm Server from receiving **send_alarm** messages.
–bell1	Sounds a single short beep for any alarm, instead of the usual distinctive tone pattern for this alarm type. The **–nobell** option overrides **–bell1**. Default if omitted: Use distinctive tone patterns for each alarm type.
–nobell	Suppresses audible alarms for all alarm types. Default if omitted: Sound audible alarms.
–p[eriod] *nnn*	Checks each alarm detector every *nnn* minutes, where *nnn* is a decimal number greater than or equal to 1. Default if omitted: Check each alarm detector every four minutes.
–v[ector] *dx* [*dy*]	Separates alarm windows by *dx* and, optionally, *dy*, where *dx* is the difference (in pixels) between the horizontal coordinates and *dy* is the difference between the vertical coordinates of the upper left corner of each alarm window. Specify *dx* and *dy* as decimal integers. For example, you might want the top left corners of successive alarm windows to be 0,0 for the first window, 20,20 for the second window, and so on; to do so, use the option **–v** 20 20. (Use the **–w** option to specify the location of the initial window.) Default if omitted: **vector** 235 0

-w[indow] *initx* [*inity* [*width* [*height*]]]

Sets the screen position of the the upper left corner of the first alarm window and the size of the alarm windows in pixels.

Default if omitted: **window** 1 1 225 100

Examples

The following example causes the Alarm Server to post a nonaudible alarm when the disk is 98 percent full (2 percent free space):

```
Command: cpo /sys/alarm/alarm_server -disk 98 -nobell
```

The following example causes the Alarm Server to post alarms when the disk is 90 percent full, when there is a new hardware failure report message, and whenever there is an observer report from the **netmain_srvr** running on node ID "ffff5." The command also sets the screen position of the first alarm window position to be near the upper left-hand corner of the screen.

```
Command: cpo /sys/alarm/alarm_server -disk 90 -hw -nm_srvr ffff5 -w 5 5
```

This next example makes use of an options file that you've created (in this case **~/user_data/opts**) to control the Alarm Server. This command also uses the **-n** option to name the Alarm Server process "alarms".

```
Command: cpo /sys/alarm/alarm_server '*~/user_data/opts' -n alarms
```

Special Considerations

Occasionally, when the the Alarm Server starts, its prints a warning message about a file called **.../alarm_server.msg_mbx**. Such a message can come from several sources. Often the problem is caused by incorrect protection of the mailbox used by the server process (see Section 3.13.2, "The Mailbox Server - **mbx_helper**"). The Alarm Server can usually change the protection on its message mailboxes automatically, but it will sometimes need your help.

The protection on two files must allow the **mbx_helper** program Read (r) and Write (w) access. Make sure that the files **'node_data/alarm_server.msg_mbx** and **~/user_data/alarm_server.msg_mbx** both allow Read and Write access to **user.server.none.**

Related Information

Information about using the Alarm Server with **netmain_srvr** is provided in the section on the **netmain_srvr** process in Appendix A.

Additional information about using the Alarm Server is available with certain optional software packages. For example, the Domain Software Engineering Environment (DSEE™) is an optional software package that uses the Alarm Server. If DSEE is installed on your system, see the DSEE manuals for additional information.

3.13.2 The Mailbox Server: mbx_helper

The Mailbox Server (**/sys/mbx/mbx_helper**) assists processes on different nodes that communicate by using mailboxes. This server must run on any node that sends or receives mailbox communications across the network.

The **mbx_helper** process is automatically started by any Apollo servers or programs that need the server to operate correctly. To send a message via the **send_alarm** command, both the sending and receiving nodes must have an **mbx_helper** process running. For the sending node, if it doesn't already have an **mbx_helper** running, the **send_alarm** command will start one. At the receiving node, the **alarm_server** process will start an **mbx_helper** process if one doesn't already exist there.

You only need to start an **mbx_helper** explicitly, with a DM **cp** command or from a start-up script, if you are running a non-Apollo server that requires **mbx_helper** to operate.

For further information about starting **mbx_helper** from programs, see *Programming With Domain/OS Calls*. To provide network services to a node, enable **mbx_helper** by removing the pound sign (#) from the appropriate start-up file on that node.

Special Considerations

In a secure network, a mailbox receives its permissions from the directory in which it is created. The ACL templates we supply ensure that server processes can have access to each other. If you do not use the templates we supply, be certain that the permissions you use allow clients on remote nodes to access a node's **mbx_helper**. If user programs have difficulty accessing **mbx_helper**, be certain that users know how to use **mbx_helper** in a secure network. See *Programming With Domain/OS System Calls* for more information.

You cannot change a mailbox's permissions while the mailbox is in use.

3.13.3 The Diskless Node Server: netman

The **netman** (**/sys/net/netman**) process manages requests from diskless nodes for access to the operating system. Diskless nodes, having no place to store the necessary operating system files, use a disked node's disks for storage.

The **netman** process, running on a disked node, receives "request for volunteer" broadcasts from diskless nodes attempting to boot. **Netman** looks in its **/sys/net/diskless_list** for a diskless node's hexadecimal ID. If the ID appears in the list, the diskless node can read, from **netboot**, the disked node on which **netman** runs. To control the distribution of disked node resources in the network, specify which diskless nodes can use a given disked node as a partner. Do this by placing diskless node IDs in the partner's **/sys/net/diskless_list**.

If a diskless node does not have its own **'node_data.***diskless.node_id* directory when it boots, **netman** will create one for it from a template in the disked node's **/sys/dm/startup_templates** directory. See Section 3.9, "Start-Up File Summary," for more information about node start-up directories.

The **netman** server runs as a background process from the a start–up file. The command line that executes **netman** is in the default start–up files that arrive with your system. Remove the pound sign (#) at the beginning of the command line to enable the process. The **netman** server will execute from the start–up script when the node is rebooted. You may start the process from the DM command line as shown in the next subsection.

> **NOTE:** We do not recommend that you run **netman** on an internet routing node. Network traffic between a diskless node and the routing node would compete with the internet traffic on the routing node, slowing both the internet traffic and the response time of the diskless node.

Starting and Stopping netman

To start **netman** from the DM command line, enter the following command:

```
Command: cps /sys/net/netman
```

The server process begins immediately and persists after logout. Another way to start **netman** is to uncomment the following line in the disked node's **/etc/rc** file:

```
# cps /sys/net/netman
```

The server process begins when the node is booted, and continues under normal conditions until it is intentionally stopped with the shell command **sigp**, as shown:

```
$ sigp netman –q
```

With both start–up methods, the process stops running if the node is shut down intentionally or if the system crashes. Restart the process if it stops running by rebooting the node or by entering the **cps** command from the DM command line.

Special Considerations

For a temporary fix, you can use **netman** if your disked node malfunctions. To do this you must shut down and reboot your node diskless. **Netman** allows any disked node to continue functioning even if its disk becomes nonfunctional.

To use **netman** when your disked node malfunctions, complete the following steps:

1. Determine the node ID of any node running **netman**. (You may want to start the **netman** process on a disked node in the same loop as the now diskless node.)

2. Use the DM command **shut** on the node that you want to boot diskless.

3. Then type the following at the Mnemonic Debugger prompt:

```
> re

> di n node_id   {node_id is the node ID of the node running netman}

> ex domain_os
  Network Partner ID nnnn
```

In this procedure, **netman** bypasses **/sys/net/diskless_list**. You don't need to edit the list to get the node back in the network. When you use this procedure, **netman** creates a **'node_data/startup[.*type*]** file for the node with disk problems.

3.13.4 The Tablet Server: sbp1

The Tablet Server (**/sys/dm/sbp1**) supports tablets that conform to the binary output mode used by Summagraphics Corporation tablets. To use a tablet, enable the Tablet Server support program in the node's **/etc/rc** file. The server process sends a control character to the tablet bit pad. An operating system program then provides the actual bit-pad support. The process started in a start-up file stops running when the operating system has taken control.

Starting the Tablet Server

To start the Tablet Server, uncomment the following line in the **/erc/rc** file:

```
# cps /sys/dm/sbp1 /dev/sio2 1
```

To use a different SIO line, change the "2" in the command to the desired number. The letter "1" indicates the mode and sampling rate selector that is sent to the tablet. You may change this mode to one of the other modes described in the *Summagraphics Corporation Bit Pad One User's Manual* (Form 64).

Special Considerations

The bit-pad support is internal to the operating system, and the server mentioned here only enables the operating system support and then stops. Therefore, the **pst** command does not show a process for the bit pad. To check on a bit-pad process, use the **tctl** command on the SIO line to which the bit pad is connected, and check that **-bp_enable** is true. The bit-pad support program is running properly if the **tctl** output contains the following line:

```
bp_enable: true
```

3.13.5 The siomonit and siologin Line Servers

Previous releases used the **siomonit** and **siologin** SIO line servers to allow you to connect a "dumb" terminal to a workstation, either directly or via modem and telephone lines from another location. Although we now recommend that you use the **/etc/ttys** file (see Subsection 3.8.5) to enable SIO lines, you can instead use **siomonit** and **siologin**.

The SIO line server processes are

- **siomonit**
  The SIO process monitor (**/sys/siomonit**). Typically invoked as a background server process from the **/etc/rc** file.

- **siologin**
  The SIO line login (**/sys/siologin**). Invoked by **siomonit**. It must be a manager within the log-in protected subsystem. **Siologin** is the process that directly manages user login.

Below, we describe the procedure used to connect terminals to workstations. Before you can use this procedure, set up the necessary configuration files and enable the server processes **siomonit** and **siologin**, described in the two subsections that follow this one.

To connect a node's SIO lines to a dumb terminal and/or modem, follow these steps:

1. Disable the SIO lines in the **/etc/ttys** file. To do so, specify **off** in the *status* field for all but the first entry in the **/etc/ttys** file.

2. Use the shell command **tctl** to display the SIO line configuration. The default values for the SIO lines are

   - 9600 baud
   - No Parity
   - Eight bits per character
   - One stop bit

3. Enable the **siomonit** server with the proper configuration files and arguments. The **siomonit** process starts the **siologin** process.

4. Connect the terminal or modem to the SIO line. Use the cable and directions supplied with the terminal or modem. If you have connected a modem, the line parameters on the terminal and modem at the remote site must match those you specified at the workstation. Connect the modem to the telephone. When you are ready to log in, signal the SIO line by pressing <RETURN> on a locally connected terminal. From a remote terminal, dial the number of the phone line connected to the node's modem.

5. When you are finished using a local or remote terminal, press CTRL/Z (or the character you have defined as the EOF character using **tctl**) to end the **siologin** process. On a local terminal, you are disconnected. On a remote terminal, you are disconnected from the node and the phone connection is broken.

3.13.5.1 The SIO Line Log-In Server: siologin

The **siologin** process uses the following command line syntax:

siologin *dev_name* [[**-dialin**] [**-n** *name*] *prog* [*argument...*]]

Each **siologin** server process waits for a carriage return character from a terminal connected directly to the SIO line, or a Data Carrier Detect (DCD) signal from a modem if the **-dialin** option has been specified. The modem generates this signal when it answers a dial-in from a remote terminal.

Upon receiving the character or signal, **siologin** invokes the operating system log-in sequence. If the sequence is successful, **siologin** logs the user in and starts a program. You specify the program with the *prog* specification. The default option starts the shell command line interpreter program **/com/sh.** *The dev_name argument*, which must be specified, is the SIO device descriptor pathname. Other options, if specified, must precede *prog* and its arguments. The **siologin** subsystem process stops when the user logs out (usually with CTRL/Z).

The **siologin** command line syntax appears as an argument list in a file used by **siomonit** (usually **'node_data/siomonit_file**). We describe the meaning of **siologin** options and arguments below.

Siologin looks for a start-up file **'node_data/startup_sio.sh** and, if it exists, executes it as a shell command file by passing it the SIO line number as an argument. For example, for **/dev/sio2:**

```
 /com/sh `node_data/startup_sio.sh 2
```

The sample file, **/sys/siologin/startup_sio.sh,** provided with the system includes the **tctl** command shown below to to ensure that the SIO line is not locked through some previous failure:

```
# ulkob /dev/sio2 ^1 -f
```

The siologin Options and Arguments

The **siologin** process also accepts the following arguments:

dev_name (required)	The SIO device descriptor pathname, either in the form **/dev/sio***x* or **/dev/Hy**0*x*, where *x* is the number of the SIO line to which the terminal or modem is connected. Use SIO line numbers 1, 2, or 3 for nodes with three SIO lines; use SIO line numbers 1 or 2 for nodes with two SIO lines. The names **/dev/sio***x* and **/dev/Hy**0*x*, where *x* is a number from 1 to 3 are synonymous except that **/dev/sio***x* always ignores data carrier detect (DCD) while **/dev/Hy**0*x* only ignores DCD when **-dialin** is not specified.
prog	A program for **siologin** to start after the login is complete. If omitted, the default invokes **/com/sh** to start the shell command line interpreter.
args	Arguments for the program specified in *prog*.

The **siologin** process accepts the following options:

-dialin	The SIO line connection is remote. If the line is remote, **siologin** asks for an access password before invoking the log-in sequence. The access password is a single string read from **'node_data/siologin_access**. For **/dev/Hy0***x*, **siologin** waits for a carrier detect signal to initiate the operating system log-in sequence. It disconnects the line after the invoked program returns. (If the connection is local, **siologin** waits for a <RETURN> before beginning the log-in sequence and signals the program connected to the line when DCD becomes unasserted.) **Siologin** logs invalid log-in attempts in the file **'node_data/siologin_log**.
-n *name*	Specifies the name to give the **siologin** process. (Takes precedence over the option **cp -n** when **siologin** is created through the DM command instead of through the **siomonit** server process.)

Special Considerations

When you use the **-dialin** option to specify remote access, you must specify an access password in the file **'node_data/siologin_access**. Create the file by entering a password as a left-justified single string in the file's top line.

Siologin logs successful and unsuccessful log-in attempts from remote terminals. It creates the file **'node_data/siologin_log**, which reports the SIDs of users who log in successfully, and provides error messages describing unsuccessful log-in attempts. You should monitor **'node_data/siologin_log** and periodically delete it, or delete old entries, since it is not self-limiting in size. Figure 3-12 shows a sample log file.

```
Thursday, February 14, 1985  13:40:52
     ** Bad (or no) access code in `node_data/siologin_access **
Thursday, February 14, 1985  13:40:52
     ** Hanging up phone **
Tuesday, February 19, 1985  15:52:29
Invalid login attempt by: jones
Tuesday, February 19, 1985  15:53:37
Invalid login attempt by: jones
Tuesday, February 19, 1985  15:53:48
    jones.dev.mktg.5de logged in
Tuesday, February 19, 1985  15:58:19
    Caller logged out.
```

Figure 3-12. Sample 'node_data/siologin_log File

To operate, the **siologin** server must have manager status in the log-in protected subsystem (see *Using Your Aegis Environment* for more information about the subsystem.) The software installation procedures give **siologin** manager status. If the server does not operate properly when **siomonit** starts it, display its subsystem status as follows:

$ subs /sys/siologin/siologin

If you receive the following message, **siologin** has the proper manager status:

```
"/sys/siologin/siologin" is a login subsystem manager
"/sys/siologin/siologin" is a file subsystem data object
```

If, however, you receive the following message, the **siologin** process has lost its manager status:

```
"/sys/siologin/siologin" is a nil subsystem manager
"/sys/siologin/siologin" is a file subsystem data object
```

To reassign manager status to **siologin,** you must log in with a **sys_admin** account and type the following:

```
$ ensubs login
$ subs /sys/siologin/siologin login -mgr
$ CTRL/Z
```

3.13.5.2 The SIO Process Monitor: siomonit

Siomonit supports repeated logins over SIO lines, independent of any log–in or log–out activity at the node terminal. To enable **siomonit,** create a file that describes the attributes of each **siologin** manager that the **siomonit** server process should start. The **siologin** processes started by **siomonit** are called its child processes.

The file passed to **siomonit** contains argument lists. Each argument list has the form of the **siologin** command line described previously. **Siomonit** invokes a separate **siologin** process for each argument list in this file. A maximum of three argument lists (one per SIO line) can be given. Comments can be included in the file with a pound sign (#).

Starting siomonit

To invoke **siomonit** from the DM command line, enter the following:

Command: **cps /sys/siologin/siomonit** *siomonit_filename*

The server process begins immediately and continues after logout.

We recommend this start–up method if you use **siomonit** or **siologin** only occasionally. This is also the way to start the process after you log in or if the process dies. Be sure the SIO lines are configured correctly by using the **tctl** command. Usually the *siomonit_filename* argument is **'node_data/siomonit_file.**

To start **siomonit** from a start–up file, uncomment the line that reads:

```
# cps /sys/siologin/siomonit -n siomonit `node_data/siomonit_file
```

Be sure this line appears after any lines in the start–up file that set environment variables.

The server process begins when the node comes online and continues across logins and logouts at the node terminal. We recommend this start-up method if you use **siomonit** and **siologin** frequently.

Signaling the siomonit Process

Sometimes you will want to signal **siomonit** once you have started it. You might do this, for example, after you make changes to an argument list or if you add a new argument list to the file **siomonit_file**.

To make **siomonit** reread the argument file and execute the process again, signal **siomonit** with an asynchronous quit fault (**sigp -q**). This is the default option of the **sigp** command:

```
$ sigp siomonit
```

The **siomonit** process goes back to its argument file and redoes whatever it finds there. Note however that **siomonit** will never stop an active child process for a given SIO line, even if you have changed the argument list for that SIO line. You must stop the child process. This will also cause **siomonit** to reread the **siomonit_file**. To stop **siomonit**, send it a stop fault (**sigp -s**).

Restarting siomonit

If the **siomonit** process stops running, restart it from the DM command line (or by rebooting the node). Check the **siomonit_log** file first, to determine why the process stopped.

Sample siomonit_file

The siomonit_file name argument lists take the form:

[-repeat] *siologin_arg_list*

The arguments in the siomonit_file are explained below.

-repeat	Configures **siomonit** to restart this **siologin** process after a user logs out. For example, when a user of an SIO line logs out, no one else can log in to that line unless this argument is in effect. It indicates that **siomonit** to restart the **siologin** process. This argument should be the first one given.
siologin_arg_list	The list of **siologin** arguments and options are described in the section on **siologin**. Arguments are passed to **siologin** unvalidated; however, the first argument must be either **/dev/sio***x or* **/dev/Hy0***x*. **Siomonit** reads the argument file over again each time a child process stops, when a user logs out, or when it receives a quit fault.

Figure 3-13 shows a sample **'node_data/siomonit_file**. You can include comments by placing the pound sign (#) at the beginning of a line.

```
# sample file for using siomonit
#
# Configure the sio lines and invoke siomonit with a line like this
# in  `node_data/startup (or /sys/dm/startup):
# cps /sys/siologin/siomonit -n siomonitor `node_data/siomonit_file
#
# Put the sio line startup file in `node_data/startup_sio.sh.
#
# and put a file like this one in `node_data for the node to be used.

# watch sio lines 1 and 2 and keep them available for siologin:
# line 1 is a dial up line:
-repeat /dev/Hy01 -dialin -n siologin1  /com/sh -f -c user_data/startup_sh
# line 2 is a local connection:
-repeat /dev/sio2   -n siologin2_local  /com/sh -f -c user_data/startup_sh
#
# up to three siologin arg lists like the ones above may be included, one
# for each sioline (only two will work for DN300's).
#
# This file is re-read by siomonit each time it re-invokes an siologin
# process or when it receives a signal via:    sigp siomonit
```

Figure 3-13. Sample 'node_data/siomonit_file File

For each argument it finds in the list, **siomonit** invokes the **siologin** program.

/sys/siologin/siologin *siologin_arg_list*

Special Considerations

Use the log files **'node_data/siomonit_log** and **'node_data/siologin_log** to help you debug **siologin** or **siomonit** server problems. Figure 3-14 shows a sample of an **siomonit_log** file.

```
Tuesday, February 19, 1985  9:28:13
    Quit fault. Restarting any dead processes...
Tuesday, February 19, 1985  11:05:56
    ** Process didn't stay alive for 15 secs:**
   /sys/siologin/siologin  /dev/siol -n siologinl
Tuesday, February 19, 1985  11:08:15
    ** Received stop fault. Closing up shop. **
Tuesday, February 19, 1985  16:34:53
   Restarted process:   /sys/siologin/siologin /dev/siol -n siologinl
Tuesday, February 19, 1985  9:18:02
 * Couldn't open command input file `node_data/siomonit_file. Status
   1010015 *
```

Figure 3-14. Sample 'node_data/siomonit_log File

If **siomonit** terminates and its child processes do not also terminate, the child processes are, in effect, orphans. The existence of the orphans interfere with **siomonit's** attempts to restart new child processes when it restarts. **Siomonit** itself never stops a child process, its own process, or an orphan process. However, **siomonit** will not be notified when an orphan process terminates. Instead, if **siomonit** detects an orphan's existence, it wakes up every 15 minutes to see if the orphan has stopped. If the orphan process has ended, **siomonit** restarts the **siologin** process as directed in its argument list. Should this occur, a user may have to wait 15 minutes before completing the **siologin** process.

Chapter 4

Creating and Maintaining the Registry

Contents

Chapter 4

Creating and Maintaining the Registry

The Domain/OS registry consists of a database of account information and associated software that enables you to store and manipulate account information, thereby controlling access to the computers on your network. The registry database can be replicated so that it resides on several nodes in a network or internet. A server must run on each of these nodes.

This chapter focuses on the **network registry**, a registry of account information that is accessible throughout your network or internet via registry servers. In addition to the network registry, you can create **local registries** residing and operating on individual nodes. Ordinarily, local registries are used only as a backup when the network registry is unavailable. Chapter 4 also provides the following information:

- Overviews of the registry and the registry server
- A discussion of network registry administration and local registries
- A description of how to operate small networks with local registries

Through the rest of this chapter, the term "registry" refers to the network registry (database and software). The term "local registry" always appears in full.

4.1 Registry Software

The registry server manages changes to the data and maintains consistency among the replicas of the registry database. Any node must communicate with a registry server when it requires access to registry information—for instance, when it checks an attempt to log in. Nodes use the Location Broker, a component of the Network Computing System™, to locate registry servers.

Figure 4–1 shows a sample network configuration, including a master registry node, a replica registry node, and normal nodes. The figure indicates where the registry server (**rgyd**) and the Location Broker daemons (**glbd** and **llbd**) are running, where replicas of the registry database reside, and where local registries reside.

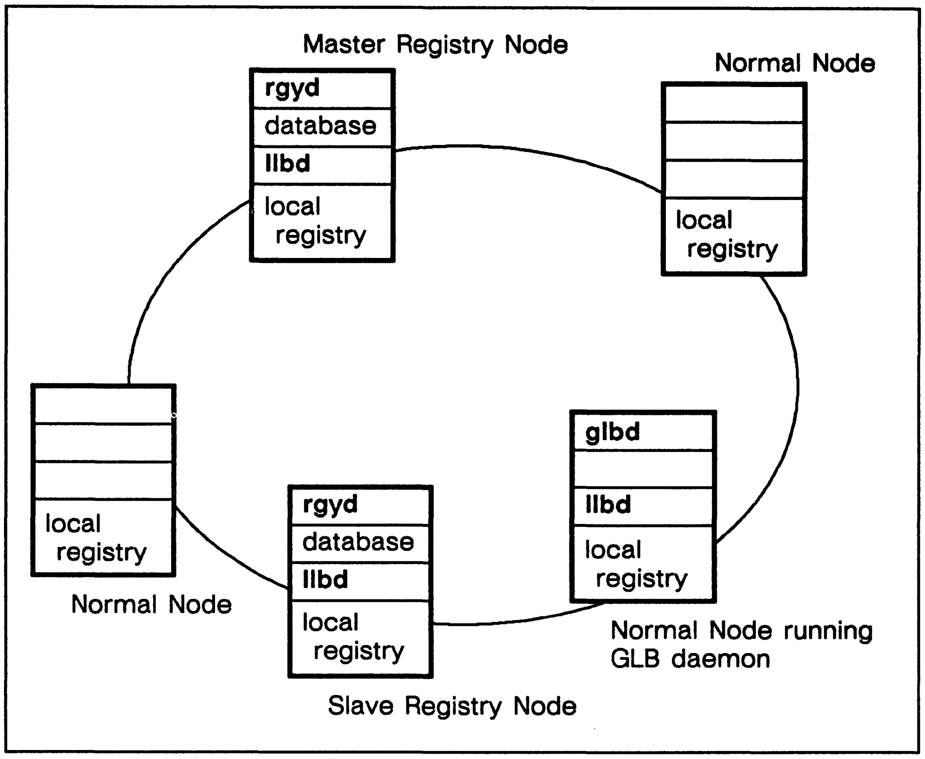


Figure 4–1. Registry Components

4.1.1 The Registry Server

The registry server, **/etc/rgyd**, can run either as a **master** or as a **slave**. There is exactly one master server on a network or internet. All other servers are slaves.

The master server handles operations that change the database (for example, adding an account or changing a password) and propagates the changes to the slaves. Either the master or a slave can handle operations that only read the database (for example, verifying a password when a user logs in). Thus, the master server must be available for any editing operation, but a slave server suffices for lookup operations. Section 4.4 describes how the servers maintain consistency among the databases.

While it runs, the registry server, whether a master or a slave, maintains its database in virtual memory. The server periodically saves the database to a copy that resides on disk. Because the copy on disk is not always up to date, there are special procedures you should follow for backing up the registry database and for restoring the database from a backup tape. These procedures are detailed in Sections 4.8, "Routine Manitenance," and 4.10, "Troubleshooting."

We recommend that, in an internet, at least one registry server exist on every network. We discuss considerations for configuring servers in Subsection 4.6.1.

4.1.2 Tools for Editing and Administering the Registry

Nearly all editing of the registry database must be done with **edrgy**. (The only exceptions are certain operations that users can perform on their own accounts, such as changing passwords.) The master registry server must be available to receive changes.

The **rgy_admin** tool administers the registry server. It can list, reset, replace, and delete replicas of the registry. It also performs special functions such as changing the master registry site and reinitializing a slave registry server.

The **cvtrgy** tool enables you to run SR10 and pre-SR10 registries simultaneously on a network or internet. See *Making the Transition to SR10 Operating System Releases.*

If you are creating a Domain internet, you can use **rgy_merge** to merge data from registries that have operated in partitions of the internet. See *Managing Domain Routing and Domain/OS in an Internet.*

If your Domain systems coexist with other UNIX systems, you should ensure that name and account information is consistent between the Domain/OS registry and the UNIX password and group files. The **import_passwd** tool, described in Section 4.11, helps you to establish consistency.

4.1.3 Location Broker Software

As we mentioned earlier, nodes use the Location Broker to locate registry servers. Two daemons, the Local Location Broker daemon (**llbd**) and the Global Location Broker daemon (**glbd**), are essential to Location Broker operation and hence to operation of the registry. A **glbd** must run on at least one node in a network. In an internet, each network must have at least one **glbd**. An **llbd** must run on all registry server nodes and on all **glbd** nodes. *Managing the NCS Location Broker* provides information about starting and running these daemons.

4.2 The Registry Database

The registry database stores information about **names**, **taccounts**, and **policies**. Table 4–1 shows the general categories of data in the registry database.

Table 4–1. Registry Database Categories

Names	**Accounts**	**Policies**
Persons	SIDs (pgo triplets)	Registry Policy
Groups (member lists)	Passwords	Organization Policy
Organizations (member lists)	Home Directories	
	Log-In Shells	

All of these registry database objects can be edited with the **edrgy** command.

4.2.1 Names

Names establish the existence of individual persons, groups, and organizations within the registry. Identical names can exist in different domains of the registry database—there can be a person, a group, and an organization with the same name.

Persons can have a **primary name** and one or more **aliases**; an alias is an alternate name for a person. Groups and organization have only primary names, not aliases.

Associated with each primary name are a **UID (Unique Identifier)** and a **UNIX number** that the operating system uses as identifiers. A primary name or alias can also have a **fullname**. The operating system doesn't use fullnames but provides them for easy recognition by users.

Unique Identifiers

The operating system associates each primary name with a (UID) consisting of a node ID and a time stamp. When printed as a text string, a UID has two parts separated by a period, as in **38edde17.50007c5f**.

The association between UID and primary name is fundamental to the protection of files and directories in your network. If you destroy this association (for example, by deleting the name from the database), you effectively "orphan" all files owned by that UID. You can use **edrgy** to "adopt" the orphaned files (that is, associate the UID with a name).

Numbers

A **number**, in the context of the registry database, is a decimal identifier associated with a primary name. Numbers are analogous to UNIX user and group IDs, so we sometimes refer to them as **UNIX numbers** or **UNIX IDs**. They exist for UNIX compatibility and are used mainly by UNIX programs.

If you convert a registry from SR9 to SR10, the conversion tool adds numbers to the registry database; if you are building a new registry database, you assign numbers when you create names. If your Apollo systems share files with other UNIX systems, you should ensure that names, UNIX IDs, and account information are consistent between the Domain/OS registry and the foreign systems. Subsection 4.6.5 contains more information about establishing uniform UNIX IDs.

Aliases

Aliases enable you to establish several accounts that have different home directories and passwords but share the same access rights to files. You can use **edrgy** to change an alias into a primary name and a primary name into an alias. Although it is possible to create an alias without an existing primary name, you ordinarily create a primary name entry before you create any aliases for that name. Groups and organizations can have only primary names, not aliases.

Fullnames

A **fullname** is a text string associated with a person, group, or organization name. The fullname typically describes or expands the name. For example, the person name **owright** could have the fullname **Orville Wright**, and the organization name **rd** could have the fullname **Research and Development.**

4.2.2 Accounts and Subject Identifiers

Accounts define who can log in to the computers on your network. An account associates a person, a group, and an organization. Each account includes information that the operating system needs when a user logs in, such as a password and a home directory.

Subject Identifiers

The operating system identifies each account by a **Subject Identifier (SID)** consisting of a person name, a group name, and an organization name, separated by periods, as in **rubinstein.pianists.none**. We use the terms **pgo** and **pgo triplet** as synonyms for SID.

In some SIDs, a percent character (%) can appear as a wildcard in place of a name. For instance, **mozart.%.%** matches **mozart.pianists.none** and **mozart.symphonists.classical**. Wildcards are not allowed in account SIDs, but they are allowed in the SIDs that specify ownership of registry data.

Other Account Information

For each account, the operating systems stores the following information in the registry database:

- An SID, for example, **grappelli.violinists.jazz**.
- An abbreviation for the SID, for example, **grappelli** or **grappelli.violinists**. At login, a user need only enter the abbreviation to specify the SID. When you add an account, **edrgy** automatically assigns the shortest unique abbreviation unless you specify otherwise. For example, if the SID **babar.elephants.none** already exists with the abbreviation **babar**, a new account for **babar.kings.none** will have the abbreviation **babar.kings**.
- An encrypted password. When a user attempts to log in, the operating system prompts for the password, encrypts the text string that the user enters, and checks the result against the encrypted string stored in the database.
- A flag that determines whether the account is valid. If an account is not valid, the operating system will reject any attempt to log in on that account.
- A home directory.
- A log–in shell.
- Miscellaneous information. This is a text string that typically describes the user of the account. If the person specified in the SID has a fullname, the person's fullname is concatenated with the account's miscellaneous information to form the *gecos* field in the **/etc/passwd** file.

4.2.3 Reserved Names and Accounts

We supply several **reserved** names and accounts with the operating system, for use by various system operations. You cannot change the UIDs and numbers associated with reserved names.

Table 4–2 lists these names and accounts.

Table 4-2. Names, Accounts, and SIDs

Person	Group	Organization	Person.Group.Organization
admin	backup	apollo	none.none.none
bin	bin	none	user.none.none
daemon	daemon	sys_org	sys_person.none.none
lp	locksmith		admin.none.none
root	login		daemon.none.none
sys_person	mail		bin.bin.none
user	none		lp.bin.none
uucp	server		uucp.daemon.none
none	sys		root.staff.none
	staff		
	sys_admin		
	sys_proj		
	wheel		

In addition to those required for reserved accounts, we supply the following memberships: the person **user** is member of the groups **backup** and **sys_admin** and the organization **apollo**; the person **bin** is a member of the group **mail**, and the person **root** is a member of the groups **bin** and **sys**.

4.2.4 Groups and Organizations

Here we describe some features that are specific to groups and organizations.

Passwords

Each group or organization can have a password. In the SysV environment, if a group has a password, a user who is not a member of the group can acquire its privileges by invoking the **newgrp** command and entering the correct password.

Many sites opt not to assign passwords for groups and organizations.

Membership Lists

Each group or organization has a **membership list** containing the person names of all its members. In order for you to create an account, the specified group and organization must contain the specified person in their membership lists.

For example, in order for you to create an account **mahler.symphonists.none,** the group **symphonists** and the organization **none** must have the person **mahler** in their membership lists. (If you are the owner of the group **symphonists** and the organization **none, edrgy** automatically adds **mahler** to their membership lists so that you can create the account in one step. If you are not the owner of **symphonists** and **none** and **mahler** is not already on these lists, **edrgy** issues an error. We describe owners of registry objects in Subsection 4.2.7.)

Project Lists

BSD UNIX systems associate each user process with not one group but a set of groups. This set consists of the group specified for the user in **/etc/passwd** and any other groups of which the user is listed as a member in **/etc/group**. The user always has the access rights that accrue from membership in every group in the set.

Domain/OS creates such a set of groups, called a **project list**, if you have the PROJLIST environment variable set to **true**. The project list consists of all groups of which the user is a member. If PROJLIST is not set, only the **pgo** of the user process controls access to files. Thus, Domain/OS allows you to choose either BSD or SysV behavior.

By default, PROJLIST is set automatically if your SYSTYPE is **bsd4.3**.

Not every group can be included in a project list. The reserved group **locksmith**, for instance, has a property that prohibits its inclusion in project lists. When you create a group with the **edrgy** command, the group is set up to allow inclusion in project lists unless you explicitly specify otherwise. If you change the properties of a group to disallow its inclusion in project lists, you cannot undo the change.

4.2.5 Policies

The registry database includes **policies** that regulate the following aspects of accounts and passwords:

- Password expiration date
- Password lifespan (how long a password remains valid before it must be changed)
- Account lifespan (how long an account remains valid)
- Minimum length of passwords
- Whether a password can consist entirely of spaces
- Whether a password can consist entirely of alphanumeric characters

You can use **edrgy** to set any of these policies for the registry as a whole or for a particular organization. If a policy is set both for the registry and for an organization, the stricter policy applies. For example, if registry policy specifies a minimum password length of six characters and policy for the **rd** organization specifies eight characters, the account **bell.comm.rd** must have a password at least eight characters long.

By default, when the registry is created, policies are at their most permissive—passwords have no expiration date, passwords and account have unlimited lifespans, there is no minimum password length, and passwords can consist entirely of spaces or entirely of alphanumeric characters. Many sites set one or more of these policies to be stricter.

4.2.6 Properties

The registry database also includes the following **properties**:

- The owner of the registry as a whole
- The owner of each name domain
- Whether UNIX restrictions are enforced
- Whether UNIX restrictions are met
- Whether the registry database is read-only

You can use **edrgy** to set any of these properties.

UNIX restrictions are met if no names exceed eight characters, all accounts have the **p** (person only) abbreviation, and all passwords are valid for vanilla UNIX systems. If UNIX restrictions are enforced, all data in the registry database must meet the restrictions. (Reserved names and accounts are excepted; they do not affect whether UNIX restrictions are met and they are not affected if the restrictions are enforced.)

4.2.7 Owners

Every object in the registry database has an owner who is allowed to modify that object. There are also owners for the registry as a whole and for each of the three name domains (person, group, and organization).

Owners are represented by SIDs. These SIDs can include wildcards.

Default Ownerships

All ownership information is contained in the registry database and can be edited via **edrgy**. When the registry is first created, the initial data, the name domains, and the registry as a whole are all owned by **%.%.%**. By default, all data added via **edrgy** is owned by **%.sys_admin.%**.

Some sites assign owners other than the defaults for registry information. You can use the **change** command in **edrgy** to assign ownerships for any names. You can use the **prop** command to assign ownerships for the three name domains and for the registry as a whole. To specify the default owner for data added via **edrgy**, issue the **defaults** command at the beginning of every **edrgy** session.

At most sites, it is simplest to have one owner for the registry and all of its data. This owner can be the unrestrictive SID **%.%.%**, a restrictive SID such as **%.sys_admin.%**, or an SID created specially for registry administration such as **%.registry_admin.%**. However, some sites prefer to have different owners for different data. A common scheme is to use organizations as administrative domains. For example, you can assign **%.sales_admin.sales** as the owner for the **sales** organization; this SID will have control over the membership list for **sales** and thereby will have control over all accounts of the form **%.%.sales**.

Rights of Owners

The following list summarizes the operations that can be performed by the owner of the registry and by owners of registry objects:

- The owner of the registry can change registry policies and properties, change the owners of name domains, and use the **rgy_admin** and **rgy_merge** tools.
- The owner of a name domain can add entries in that domain. For instance, the owner of the person domain can add persons.
- The owner of an organization can add or delete members, change properties such as the password and the fullname, change policies for that particular organization, or delete the organization. If the owner of *organization* deletes *person* from the membership list, any accounts for *person.%.organization* are also deleted. If the owner deletes *organization* itself, accounts for *%.%.organization* are deleted.
- The owner of a group can add or delete members, change properties such as the password and the fullname, or delete the group. If the owner of *group* deletes *person* from the membership list, accounts for *person.group.%* are also deleted. If the owner deletes *group* itself, accounts for *%.group.%* are deleted.
- The owner of a person can change properties such as the fullname, delete the person, or add an account (provided the person is a member of the specified group and organization). Deleting *person* also deletes accounts for *person.%.%*.
- Account information such as the password and fullname can be changed by any user of the account.
- The owner of any registry object can change the owner of that object. (An owner can actually "give away" ownership.) This also applies to ownership of the registry as a whole.

Any of these registry operations can also be performed by a **root.%.%** or **%.locksmith.%** account, provided the user is logged in at the master registry node.

4.3 The /etc/passwd, /etc/group, and /etc/org Files

UNIX systems store information about accounts in the files **/etc/passwd** and **/etc/group**. Domain/OS provides these files and, in addition, an **/etc/org** for information about organizations.

In Domain/OS, the **passwd**, **group**, and **org** files cannot be directly edited. Changes to the registry database are made with **edrgy** or with specialized utilities such as **/com/chpass** and **/bin/passwd**. The registry server reflects all changes in the **passwd**, **group**, and **org** files as well as in the registry database. The server updates these files periodically, whenever it saves its database from memory to disk.

Each registry server, whether master or slave, maintains its own versions of the **passwd**, **group**, and **org** files in its copy of the registry database. On all nodes, the files in **/etc** are specially typed file system objects; when a user requests access to one of the files, the operating system finds a version of the file at one of the registry server nodes.

UNIX system calls such as **getpwent**(3) access the version of the registry database that resides in memory at a registry server node. These calls do not use the **passwd**, **group**, and **org** files at all. Thus, though the files in **/etc** are not always current with the registry database in memory, the asynchrony is harmless.

The rest of this section describes the format of the **passwd**, **group**, and **org** files in Domain/OS. See also the reference documentation for **passwd**(5), **group**(5), and **org**(5).

4.3.1 The /etc/passwd File

The **passwd** file contains one line of text for each account in the registry database. Colons divide each line into seven fields, as follows:

account–name:*encrypted–password*:*user–ID*:*group–ID*:*gecos*:*home–directory*:*login–shell*

The *account–name* is the SID of the account, abbreviated if possible. The *user–ID* and *group–ID* are the UNIX numbers of the person and group in the account SID. The *gecos* field is the concatenation of the person's fullname and the account's miscellaneous information. (Subsection 4.2.2 explains the SID, miscellaneous information, and other account data.)

The **passwd** file can include several entries for one person if the person has several accounts, as in the following excerpt:

```
arnold:QB4mmrONjs/szD:15586:98:Ken Arnold://anarres/arnold:/bin/csh
arnold.unix:ZcdWqJGsemkYJn:15586:38:Ken Arnold://anarres/arnold:/bin/csh
```

If an account is invalid, its entry in the **passwd** file will contain the text string ***NOACCT*** where the encrypted password would ordinarily appear.

4.3.2 The /etc/group File

The **group** file contains one line of text for each group in the registry database. Colons divide each line into four fields, as follows:

group–name:*encrypted–password*:*group–ID*:*membership–list*

The *group–ID* is the UNIX number of the group. The fourth field is a list of person names, separated by commas. Thus, part of a **group** file could look like this:

```
dds::78:emartin,pato,phl,pjl,mishkin,molson,mk
debug::211:gordon,cas,gkk
demo::68:
```

In the **group** file, each entry occupies one line of text.

4.3.3 The /etc/org File

The **org** file contains one line of text for each organization in the registry database. Its format is the same as that of the **group** file:

org–name:*encrypted–password*:*org–ID*:*membership–list*

The *org–ID* is the UNIX number of the org. Part of an **org** file could look like this:

```
intl::9:jimk
legal::10:barker_c,olesen_d,ponte_l,fazio_p,rossini_r,peter,\
    lewis_j,dacier_p,lynch_p,egoldman,barbara
```

In the **org** file, a backslash (\) at the end of a line indicates that the membership list continues on the next line.

4.4 How the Registry Database is Replicated

As we mentioned in Section 4.1, the registry database can be replicated. In addition to the database managed by the master server, you can create database replicas managed by slave servers. Replication of the registry database can enhance performance and reliability, especially in an internet.

The registry uses "weakly consistent" data replication: the data in every replica of the database are always available, though the data in slave databases are not always up to date. The replication mechanism ensures that all replicas of the database will converge to the same set of information while remaining available for lookup by system software and by applications.

Figure 4–2 shows part of a replicated registry configuration. The master server and one of the slave servers are pictured. Each server manages a database of name and account information and a **replica list** containing the network address of each host that runs a registry server. The master server also manages a **propagation queue** containing information that it must propagate to the slaves.

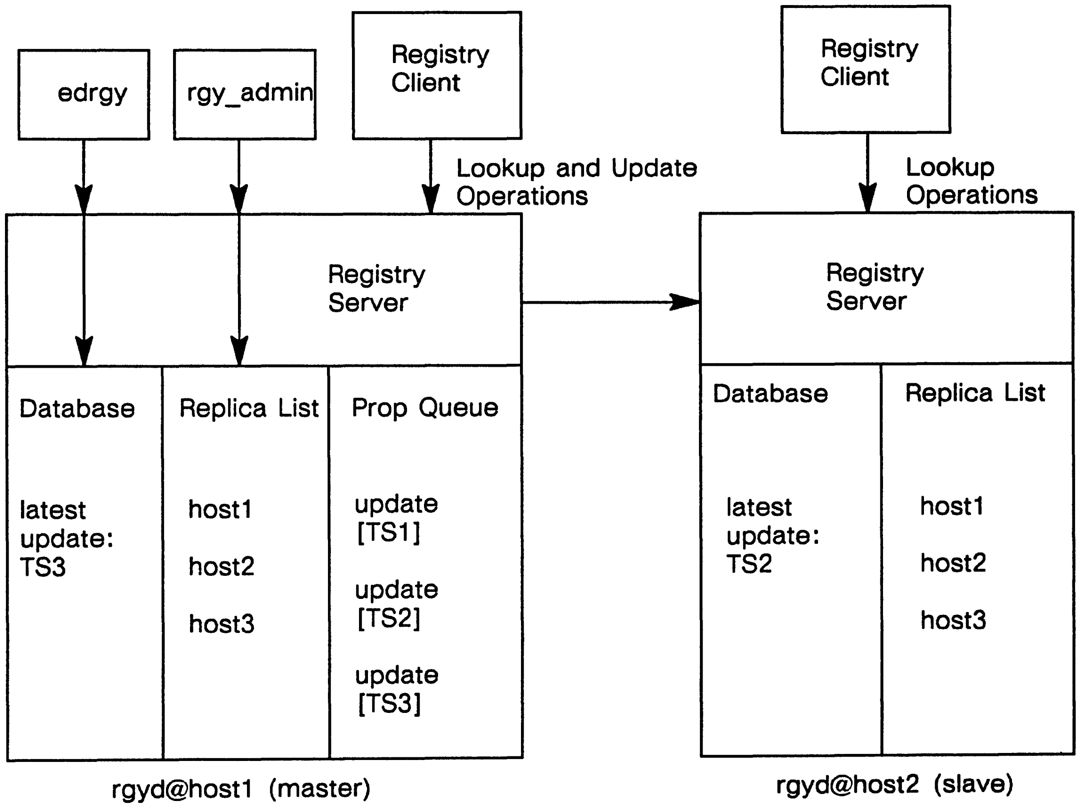


Figure 4–2. Registry Server Operation

As the figure shows, **edrgy** makes changes only to the database, and **rgy_admin** makes changes only to the replica list; these changes can be made only through the master server.

When you change any information in the database or in the replica list (for example, by adding an account, or adding a replica) the master server enters the changed information, called an **update**, in its copy of the database. It marks both the database and the update with a time stamp.

The master server places the update in the propagation queue and attempts to propagate the update to each slave server on its replica list. If propagation of an update to one of the slaves does not succeed on the first attempt, the master server retries periodically until it succeeds. The master server always propagates updates in chronological order, according to their time stamps. It keeps track of which updates are pending propagation to which slaves. When an update has propagated to all the slaves, it is removed from the propagation queue.

In Figure 4–2, for example, the update with time stamp **TS3** awaits propagation to the slave server at **host2**. Propagation of the updates with time stamps **TS1**, **TS2**, and **TS3** to the slave server at **host3**, not shown, is still pending.

If the master server loses communication with a slave server for a long time, the master server will reinitialize the slave server, giving it a fresh copy of the entire database, when communication is restored.

4.5 How the Registry Database is Stored on Disk

Each registry server, whether master or slave, maintains a working copy of its database in virtual memory and a permanent copy on disk. All lookups and updates operate on the copy in virtual memory. The server uses the copy on disk to initialize the copy in memory when it starts up.

When a master server receives an update from a utility such as **edrgy** or when a slave server receives an update from the master, the server applies the update to its database in virtual memory and also saves the update in a log file on disk. Updates accumulate in this log file in chronological order. When a server restarts (for example, when a server node boots), it initializes its database in memory from the database on disk and then it "replays" the updates from the log file. This mechanism ensures that no updates are lost when a server node is shut down.

Each registry server periodically saves its entire database from virtual memory to disk. The database is stored in the directory **/sys/registry**. When it performs a disk save, the server clears the log file.

Sections 4.8, "Routine Maintenance" and 4.10, "Troubleshooting," give procedures for backing up the registry database and for restoring the database from a backup tape.

4.6 Setting Up the Registry

This section describes how to set up the registry. Note that several of the procedures in this section must be performed by the **root** user.

In Subsection 4.6.3, you must choose one of two procedures, depending on whether you are creating a new network of nodes running SR10 or updating an existing network from SR9 to SR10.

The procedures in Subsections 4.6.5 and 4.6.8 might not be necessary at your site. All other procedures in this section are essential to successful operation of the registry.

If you are creating a Domain internet, you should see *Managing Domain Routing and Domain/OS in an Internet* for more information.

4.6.1 Planning a Configuration

Use the following considerations to plan a configuration of registry servers for your site:

- You can run more than one **rgyd** on a network. In an internet, we recommend that you run at least one **rgyd** in every network.
- Nodes that run **rgyd** should have local disks. They also should have at least 3 MB of physical memory, more for very large registries.
- Nodes that run **rgyd** should be running and available all the time. It is especially important that the node where the master server runs be available throughout the network or internet.

Choose a site for the master server. If you decide to run several servers, choose sites for the slave servers.

4.6.2 Starting Location Brokers

Before you run registry servers, you must establish Location Brokers on your network. At least one Global Location Broker (**glbd**) must run on your network. In an internet, at least one **glbd** must run on each network of the internet. A Local Location Broker (**llbd**) must run on every node that runs a **glbd** and on every node that will run a registry server (master or slave).

Managing the NCS Location Broker gives procedures for starting up Location Brokers. Refer to this manual now and then continue with Subsection 4.6.3.

4.6.3 Creating the Registry Database

There are two ways to create an SR10 registry database.

Updating an SR9 Registry to SR10

If you are updating from SR9 to SR10 system software on your network or internet, you should convert your SR9 registry database to SR10 format via the **cvtrgy** utility. Skip the rest of this subsection, follow the conversion procedures described in *Making the Transition,* then proceed with Subsection 4.6.4.

Setting Up a New SR10 Registry

If you are setting up a new network or internet of Domain nodes and SR10 Domain/OS is the only system software any of these nodes have ever run, you use the **rgy_create** utility to create a new database and initialize it with reserved names and accounts. You typically create the SR10 registry as part of your first SR10 installation. Since it is used only once, **rgy_create** resides in the **/install/tools** directory and is not included in subsequent SR10 installations. See *Installing Software with Apollo's Release and Installation Tools* for more information about **rgy_create**.

4.6.4 Starting the Master Registry Server

Log in on the master registry site node and use **/bin/ps** or **/com/pst** to verify that an **llbd** is running on the node.

To start the master server, complete the following steps:

1. Become **root** and enter the following command on the master node:

   ```
   $ /etc/server -p /etc/rgyd &
   ```

   The **-p** option causes **/etc/server** to create a process that runs under the SID that invoked the command, rather than the default of **user.server.none**.

2. Create the file **/etc/daemons/rgyd** on that node, so that the server will start each time the system boots. Use the UNIX **touch** command or the Aegis **crf** command:

   ```
   $ touch /etc/daemons/rgyd        (UNIX environments)

   $ crf /etc/daemons/rgyd          (Aegis)
   ```

4.6.5 Establishing Uniform UNIX Numbers

If your Apollo systems share files with other UNIX systems, you should ensure that names, UNIX IDs, and account information are consistent between the Domain/OS registry and the foreign **passwd** and **group** files. For the purposes of this discussion, "file sharing" can take the form of direct access through facilities such as Domain NFS or indirect file transfer via media such as **tar** tapes.

We provide a tool called **import_passwd** that helps you to identify and resolve conflicts of names, UNIX IDs and account information. If you plan to share files between Apollo systems and other UNIX systems, we recommend that you run **import_passwd** now to minimize the number of changes you have to make. Typically, you run **import_passwd** in a mode that changes IDs in the Apollo registry to match the IDs on the foreign systems; afterward, you will have to run another tool called **syncids** to ensure that the IDs stored in Apollo file systems match those stored in the registry.

See Section 4.11 for detailed information about **import_passwd**.

4.6.6 Setting Policies, Properties, and Passwords

Before you make any further changes to the registry database, you should use the **prop** command in **edrgy** to view policies and properties and change them as desired.

We provide default passwords for all reserved accounts except **user.none.none**, which by default has no password. These defaults are set either by **cvtrgy** (if you converted from an SR9 registry) or by **rgy_create** (if you created a new SR10 registry). You should use **edrgy** to change or set the passwords for these accounts.

Now is also the best time to change the owners of reserved registry data, if you do not want to use the defaults. See Subsection 4.2.7 for details.

4.6.7 Adding Names and Accounts

Your registry now contains the following names and accounts:

- Those added as reserved information by **rgy_create** or **cvtrgy**
- If you used **cvtrgy**, those derived from your SR9 registry
- If you used **import_passwd**, those derived from the password and group files on your other UNIX systems

Use **edrgy** to add any other names and accounts that your site requires. You can do this now or at any time later.

4.6.8 Creating Slave Registry Replicas

Follow the procedures in this subsection if you want to replicate the registry database.

1. Log in on a slave node and use **/bin/ps** or **/com/pst** to verify that an **llbd** is running on the node.

2. To start the slave server, become **root** and enter the following command on the slave node:

   ```
   $ /etc/server -p /etc/rgyd -create &
   ```

   This command locates the master server, adds the local node to the master replica list, and causes the master to initialize a new database at the local node.

3. As you did on the master node, create the file **/etc/daemons/rgyd**, so that the server will start each time the system boots. Use **touch** or **crf**:

   ```
   $ touch /etc/daemons/rgyd        (UNIX environments)

   $ crf /etc/daemons/rgyd          (Aegis)
   ```

4. Repeat the above steps for each replica you want to create.

   You can use the **lrep -state** command in the **rgy_admin** tool to verify that the master and slave servers are running.

4.6.9 Restarting Registry Servers

To restart a registry server, whether a master or a slave, run **rgyd** without any options:

```
$ /etc/server -p /etc/rgyd&
```

4.7 Managing the Registry

Managing the registry requires you to administer both the information in the registry database and the operation of the registry server. You use **rgy_admin** to administer the operation of the registry server and **edrgy** to edit data in the registry database.

4.7.1 Managing the Registry Server

The **rgy_admin** tool manages the operation and replication of the registry server. You can use **rgy_admin** to read and edit replica lists, delete replica registries, and stop registry servers.

When invoked, **rgy_admin** issues a prompt, **rgy_admin:**, and enters an interactive mode, waiting for your input. Sections 4.8, 4.9 and 4.10 contain procedures for some tasks you might need to perform with **rgy_admin**.

4.7.2 Editing the Network Registry Database

The **edrgy** command manages the name and account information in the registry database. Using **edrgy**, you can add, edit, or delete any of the following information:

- Information about persons, groups, and organizations
- Group and organization membership lists
- Policies for organizations and for the registry as a whole
- Properties of the registry
- Account information
- Owner information

You can also use **edrgy** to view registry database information without making changes.

4.7.3 Editing the Local Registry Database

You can use the **-l** option of **edrgy** to edit a node's local registry information. This form of the command is available to ordinary users and not reserved to the registry owner. Remember, however, that at each login, the operating system updates any entry in the local registry for the account used.

4.7.4 Merging Disjoint Registries

Network reconfigurations sometimes require you to merge the databases of two registries that have been operating independently.

We provide the **rgy_merge** tool to help you merge registry databases. This tool compares two master registry databases and informs you if any names or numbers conflict. You must use **edrgy** to resolve any conflicts before **rgy_merge** will actually perform the merge.

There are two types of reconfigurations that require merging of registries:

- Connecting two networks to create an internet
- Combining two networks into one larger network

Note that in this context, a "network" can consist either of several nodes or of a single standalone node running a network registry.

If you are creating a Domain internet, see *Managing Domain Routing and Domain/OS in an Internet*. This book describes all aspects of creating an internet and includes procedures to merge registries.

If you are combining two networks into one larger network, the procedures for merging registries in *Managing Domain Routing and Domain/OS in an Internet* still apply. You should perform the merge during off hours with as few users on the networks as possible. Until the merge is complete, two disjoint registries will exist on one network, so the chances of obtaining incorrect information in a registry lookup are greater than in the internet case.

Section 4.9 contains several procedures for dealing with network reconfigurations.

4.8 Routine Maintenance Procedures

This section contains procedures for routine maintenance of the registry.

4.8.1 Backing Up the Registry Database

Because the registry server maintains its most current data in memory and saves data to disk only periodically, we provide a special procedure for backing up the database. You should follow Procedure 4–1 either when you back up the disk containing the master replica of the database or when you back up the database only. The server will not accept updates while the backup is in progress.

Procedure 4–1. *Backing Up the Registry Database*

You must be an owner of the registry to perform Tasks 1 and 3 of this procedure.

Task 1: Put the master server in maintenance state

Use the **state** command in **rgy_admin** to put the master server in maintenance state. While in maintenance state, the server will refuse updates.

Task 2: Back up the master replica

Back up the master replica of the registry database by backing up either the entire volume or the **/sys/registry** tree.

Task 3: Take the master server out of maintenace state

Use the **state** command in **rgy_admin** to take the master server out of maintenance state. The server will resume accepting updates.

4.8.2 Checking Consistency of Registry Replicas

You can use Procedure 4–2 to check that all replicas of the registry are operating and that all copies of the database are up to date.

Procedure 4–2. *Checking Consistency of Registry Replicas*

Task 1: **Invoke rgy_admin**

Task 2: **Use the lrep –state command**

The **lrep –state** command reads the replica list at the **rgy_admin** default host, queries each replica in the list, and displays information about the each replica that responds. The output of **lrep –state** shows which server is the master, the state of each server, and the time stamp of the most recent update at each replica of the database.

In the following example, **rgy_admin** queries the slave registry server at **//tosca**, which provides information about the master replica at **//aida** and about slave replicas at **//tosca** and **//lulu**. All replicas are operating normally ("in service") and both slaves are in synch with the master.

```
$ rgy_admin
Default object: rgy  default host: dds://tosca
State: in service  slave
rgy_admin: lrep -state
dds://tosca          state: in service          1988/04/25.10:49:23
dds://lulu           state: in service          1988/04/25.10:49:23
(master)dds://aida   state: in service          1988/04/25.10:49:23
```

The online and printed reference documentation for **rgy_admin** lists and describes other states that **lrep** may report.

4.8.3 Restarting Registry Servers

If you created the file **/etc/daemons/rgyd** at every registry site, as described in Subsection 4.6.8, a registry server should automatically restart whenever the node boots. In the event that a server fails to restart or dies, you can use Procedure 4–3 to restart it by hand. This procedure applies to either master or slave servers.

Procedure 4–3. *Restarting a Registry Server*

Task 1: Become root

Become **root** on the node where you want to restart the server. Issue the following shell command:

```
$ /etc/server -p /etc/rgyd &
```

Task 2: Verify that the server is running

Use the **lrep –state** command in **rgy_admin**, as described in Procedure 4–2, to verify that the server is now running. If it is not, see the troubleshooting procedures in Section 4.10.

Task 3: Verify that /etc/daemons/rgyd exists

Check that the file **/etc/daemons/rgyd** exists so that the server will automatically restart when the system boots.

4.9 Reconfiguration Procedures

This section describes procedures to help you deal with hardware and network reconfigurations.

4.9.1 Changing the Network Address of a Registry Site

If you reconfigure a network or relocate a node that runs a registry server, the network address of a registry site may change. Since registry replicas use their network addresses to locate and identify each other, it is crucial that every replica learn of any changes to network addresses.

If the network address of a registry site has changed, you must stop the registry server at that site (via the **stop** command in **rgy_admin**) and then restart the server (as described in Procedure 4–3). The registry server automatically learns its new network address when it restarts.

Changes of network addresses can be propagated to registry replicas in three ways:

- If only the master replica changes its network address, and all slaves retain their addresses, the master will automatically propagate its new address to all slaves.
- If the master replica does not change its network address, but one or more slaves do, each slave will locate the master and give the master its new address. The master will then propagate the new address to all other slaves.
- If the master replica and one or more slaves change their network addresses, there will be no way for the master and the changed slaves to contact each other, and you should execute Procedure 4–4 to update the replica list at the master.

Procedure 4–4. *Updating the Replica List at the Master Registry*

You must be an owner of the registry in order to perform Task 3 of this procedure.

Task 1: Set the default host to the master registry site

Invoke **rgy_admin**. Set the default host to the master registry site by issuing the **set –m** command.

Task 2: List the addresses of all registry replicas

Use the **lrep –na** command in **rgy_admin** to list the network addresses of all registry replicas (as stored in the replica list at the master site).

Task 3: Update the replica addresses

For any slaves whose network addresses have changed, use the **reprep** command in **rgy_admin** to update these addresses. Use **lrep –na** again to verify that the replica list at the master site is now correct.

Once the replica list at the master site is updated, the master server will propagate changes of network address to the slave servers as necessary.

4.9.2 Changing the Master Registry Site

Smooth operation of the registry requires that the node running the master registry server always be available. If you are planning to remove this node from your network or to shut it down for an extended period, you should change the master registry site.

Procedure 4–5 describes a clean, graceful method for changing the master registry site by causing the master site and a slave site to reverse roles. It assumes that the registry servers at these two sites are operating normally. After the original master has been changed to a slave, you can use Procedure 4–6 to delete it.

Section 4.10 contains procedures for dealing with abnormal situations arising, for instance, from hardware or network failure.

Procedure 4–5. *Changing the Master Registry Site*

You must be an owner of the registry in order to perform Task 3 of this procedure.

Task 1: **Choose the master registry site**

Choose the new master site. A slave replica must exist at this site. If necessary, create the slave replica, as described in Subsection 4.6.8.

Task 2: **Set the default host to the master registry site**

Invoke **rgy_admin**. Set the default host to the current master registry site by issuing the **set –m** command.

Task 3: **Change the master registry site**

Change the master site by issuing the following command:

```
rgy_admin: change_master -to //newmaster
```

where *//newmaster* is the new master site you chose in Task 1.

Task 4: **Verify that the master site has been changed**

Use **lrep –state** to verify that the master site has been changed.

The former master site is now a slave site. You can delete this slave replica by executing Procedure 4–6.

4.9.3 Deleting a Slave Registry Replica

If you are planning to remove a slave site node from your network or to shut it down for an extended period, you should delete the registry replica at that site by following Procedure 4–6.

Procedure 4–6. *Deleting a Slave Registry Replica*

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: Set the host to the current master registry site

Invoke **rgy_admin**. Set the default host to the current master registry site by issuing the **set –m** command.

Task 2: Delete the slave replica

Delete the slave replica by issuing the following **rgy_admin** command:

```
rgy_admin: delrep //slave
```

where *//slave* is the site of the slave replica to be deleted. The master server will instruct the slave replica to delete itself.

Task 3: Verify that the slave has been deleted

Use **lrep –state** to verify that the slave replica has been deleted. The deletion may take a while. Until the slave is actually deleted, **lrep –state** will show the slave as marked for deletion.

4.10 Troubleshooting Procedures

This section contains procedures for troubleshooting the registry. You should not need to use any of these procedures except when network or hardware failures have disrupted operation of the registry.

4.10.1 Recreating a Slave Registry Replica

If a slave replica of the registry database has become corrupted or irrecoverably out of date, you can use Procedure 4-7 to recreate the replica.

Procedure 4-7. *Restoring a Slave Database and Restarting Its Server*

Task 1: **Become root**

Become **root** on the node where you want to recreate the slave replica. Use the UNIX **ps** command or the Aegis **pst** command to check whether a **rgyd** is running. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry in order to use the **stop** command.)

Task 2: **Recreate the slave replica**

Recreate the slave replica by issuing the following shell command:

```
$ /etc/server -p /etc/rgyd -recreate &
```

This destroys the existing database, creates a new one, and starts a slave server.

Task 3: **Verify that /etc/daemons/rgyd exists**

Check that the file **/etc/daemons/rgyd** exists, so that the server will automatically restart when the system boots.

4.10.2 Recovering from Loss of the Master Registry Replica

If the master registry replica becomes inoperable or unavailable for an extended time, you may need to replace it or restore it. This subsection contains several procedures for recovery of the master replica. You should read through all of them and select the one most appropriate for your particular situation.

If there exists a slave replica with a nearly current database, you can use Procedure 4–8 to turn this replica into the master. (You can check the time stamps of slave databases via the **lrep -state** command in **rgy_admin**.) If no such slave exists, you can use Procedure 4–10 or Procedure 4–11 to restore the registry database from a backup tape.

Procedure 4–8. *Turning a Slave into a Master*

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: Choose the slave replica

Choose the slave replica that will become the new master.

Task 2: Make the chosen slave the master registry site

Invoke **rgy_admin**. Issue the following **rgy_admin** commands to set the chosen slave site as the default host and to turn this slave into a master:

```
rgy_admin: set -h //newmaster
rgy_admin: become -master
```

where *//newmaster* is the site of the slave replica you chose in Task 1.

Task 3: Verify the new master site

Use **lrep -state** to verify that *//newmaster* is now the master registry site.

If the old master replica becomes available again, you must immediately change either the old master or *//newmaster* to a slave replica, to prevent inconsistencies in the database replicas. Retain the master with the more current database and use Procedure 4–9 to turn the other master into a slave.

Procedure 4–9. *Turning a Master into a Slave*

Use this procedure to turn a master replica into a slave. You should use this procedure only if you have more than one master running on your network or internet, a highly unusual condition. (If you are performing a merge of two disjoint registries, two masters will coexist temporarily, but **rgy_merge** will turn one of them into a slave.)

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: **Choose the master replica**

Choose the master replica that will become a slave.

Task 2: **Make the master into a slave replica**

Invoke **rgy_admin**. Issue the following **rgy_admin** commands to set the chosen master site as the default host and to turn this master into a slave:

```
rgy_admin: set -h //newslave
rgy_admin: become -slave
```

where *//newslave* is the site of the master replica you chose in Task 1.

Task 3: **Verify that the master is a slave**

Use **lrep –state** to verify that *//newslave* is now a slave registry site.

Procedure 4–10. *Restoring a Master Server at Its Original Site*

Use this procedure to restore a master replica at its original location. See Procedure 4–11 if you are restoring the database at a different location.

Task 1: **Become root and stop rgyd**

Become **root** on the master site node. Use the UNIX **ps** command or the Aegis **pst** command to check whether a **rgyd** is running. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry in order to use the **stop** command.)

Task 2: **Restore /sys/registry**

Restore the /sys/**registry** tree from the backup media.

Task 3: **Restart rgyd**

Restart the server by invoking **rgyd** without any options:

```
$ /etc/server -p /etc/rgyd &
```

Task 4: **Verify that /etc/daemons/rgyd exists**

Check that the file **/etc/daemons/rgyd** exists, so that the server will automatically restart when the system boots.

Procedure 4–11. *Restoring a Master Server at a New Site*

Use this procedure to restore a master replica at a new location. You must be an owner of the registry in order to perform Task 4 of this procedure.

Task 1: **Become root and stop rgyd**

Become **root** on the new master site node. Use the UNIX **ps** command or the Aegis **pst** command to check whether a **rgyd** is running at the new site. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry in order to use the **stop** command.)

Task 2: **Restore /sys/registry**

Restore the **/sys/registry** tree from the backup media. (Alternatively, if the disk that contained the master replica on the original site is still intact, you can move the disk to the new node; in this case, you must run the **chuvol** utility on the volume you are moving.)

Task 3: **Restart rgyd**

Restart the server by invoking **rgyd** with the **–restore_master** option:

```
$ /etc/server -p /etc/rgyd -restore_master &
```

Task 4: **Delete the old master site**

Use the **delrep –force** command in **rgy_admin** to delete the old master site from the replica list, as described in Procedure 4–12.

Task 5: **Create /etc/daemons/rgyd**

Create the file **/etc/daemons/rgyd** at the new site, so that the server will automatically restart when the system boots.

4.10.3 Forcibly Deleting a Registry Replica

Use Procedure 4–12 to delete a registry replica when the ordinary method described in Procedure 4–6 has failed.

Procedure 4–12 uses the drastic **delrep –force** command in **rgy_admin** to delete the registry replica from the replica lists at all other registry sites. It does not actually stop the server or delete its database and, indeed, does not communicate at all with the server. You should apply this method only if the replica has died irrevocably.

If you have forcibly deleted a replica and the replica later resumes operation, you should use the **reset** command in **rgy_admin** to stop the server and delete its database, since the forced deletion has made all other registry replicas unaware of its existence.

Procedure 4–12. *Forcibly Deleting a Registry Replica*

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: **Set the default host to the current master site**

Invoke **rgy_admin**. Set the default host to the current master registry site by issuing the **set –m** command.

Task 2: **Delete the replica**

Delete the replica by issuing the following **rgy_admin** command:

`rgy_admin:` **delrep** ***//regsite*** **–force**

where *//regsite* is the site of the replica to be deleted.

4.11 The import_passwd Command

The **import_passwd** command creates entries in the Apollo registry based on information in UNIX password and group files. It provides a method of ensuring consistency between the Apollo registry and the UNIX password and group entries. Consistency is crucial for most forms of file sharing between Apollo systems and other UNIX systems.

Use this command when you are

- Attaching Apollo node(s) to a foreign network
- Attaching foreign node(s) to an Apollo network
- Connecting Apollo and foreign networks

4.11.1 How import_passwd Processes

When **import_passwd** processes, it compares the foreign group and password file entries to the Apollo registry entries. It can find two types of conflicts:

- **Name Conflicts.** These conflicts arise when the same name string is defined in the Apollo registry and the foreign system. "Joe 102" and "Joe 555" are an example of such a conflict. The duplicate name may represent the same user or two different users.
- **UNIX ID Conflicts.** These conflicts arise when the same UNIX ID is defined in the Apollo registry and the foreign system for users with different names. "Joe 102" and "Ann 102" are an example of such a conflict.

These conflicts can be found separately, as in the examples above, or together. For instance, an Apollo entry of "Joe 102" and a foreign entry of "Joe 102" are in conflict. Unless they represent the same user, one of the entries must be changed.

As **import_passwd** processes, it performs the following steps in sequence:

1. It puts the Apollo registry in maintenance mode and reads the foreign group and password files.
2. It compares the foreign group file entries to the Apollo group entries. If there are no conflicts, it creates Apollo group registry entries corresponding to the foreign groups. (Section 4.11.3 describes what happens if there are conflicts.) Note that the members of the groups are not added at this time, but in Step 4.
3. It compares the entries in the foreign password file to the Apollo person and account entries. Again, if there are no conflicts, it creates Apollo person and account entries corresponding to the foreign file.
4. If there are members in the foreign groups handled in Step 2, it adds them to the appropriate group in the Apollo registry.
5. It then prompts for whether or not to make the changes. If you specify that the changes should be made, it saves the changes to the registry.

4.11.2 Other Entries Created by **import_passwd**

The **import_passwd** tool modifies only person names, person IDs, group names, group IDs, group members, and account passwords. It does not modify any of the additional information in the registry.

For example, assume you have a foreign password entry for user **jack** and group **staff** and an Apollo account entry of **jack.staff.none**. You run **import_passwd** with the **-i** option. This option tells **import_passwd** to consider the entries identical. The home directory specified in the foreign network is **/usr/u/jack**; the home directory specified in the Apollo network is **//gimli/jack**. The **import_passwd** tool will not change the Apollo registry to match the foreign home directory. The **jack.staff.none** entry in the Apollo registry will have a home directory of **//gimli/jack**, not **/usr/u/jack**.

If **jack.staff.none** did not exist in the Apollo registry, **import_passwd** would create a new registry entry. For the additional information, it assigns the following values:

For Person and Group Entries:

- **fullname** = " (that is, empty).
- **owner** = Same as the owner of the domain as listed in the registry properties (that is, the owner for new person entries is set to Person Owner, and the owner for new group entries is set to Group Owner.)
- **alias/primary** = Primary for first entry; alias for subsequent ones.
- **projlist_ok** (for groups only) = Yes.
- **passwd** = For groups only, taken from the foreign group file.
- **membership list** = For new groups only, all persons listed in the foreign group file and all persons with accounts in the foreign password file with that group.

For Account Entries:

- **abbreviation** = Shortest possible abbreviation that does not conflict with pre–existing Apollo accounts.
- **account_valid** = True.
- **gecos** = Same as foreign password file.
- **homedir** = Same as foreign password file.
- **shell** = Same as foreign password file.
- **passwd** = Same as foreign password file. Note that you must modify or reset imported passwords before user authentication is possible and for the account to be usable in a pre–SR10 registry.
- **passwd_dtm** = Date and time **import_passwd** was run.
- **passwd_valid** = True.

4.11.3 Resolving Conflicts

When you use **import_passwd**, you must decide how to resolve the conflicts it will encounter. The **import_passwd** command provides a number of options to help you. If the conflict cannot be resolved even with the option instructions, **import_passwd** will prompt you for resolution. The options are described in the following subsections.

The Identical User Option

The **–i** option lets you specify that duplicate names are not in conflict but, instead, represent the same identity. When **import_passwd** finds duplicate name entries, it processes them as though they are the same user. If you do not use the **–i** option, **import_passwd** will prompt you to resolve the name conflict.

The Favored Entry Option

The **–a** (favor Apollo) and **–f** (favor foreign) options let you specify whether the Apollo or the foreign entry is the favored entry. A favored entry is retained as is. You are prompted to modify non–favored entries.

For example, suppose you run **import_passwd** with the **–a** (favor Apollo) option and without the **–i** (identical user) option. During processing, the program encounters an Apollo entry of **joe 555** and a foreign entry of **joe 102**. Because the Apollo entry is favored, **joe 555** will be retained in the Apollo registry, and you will be prompted for a new UNIX ID (a new name) for **joe 102**.

The **import_passwd** command also uses the favored entry to resolve UNIX ID conflicts. For example, suppose you are running **import_passwd** with the options described above. During processing, it encounters an Apollo entry of **joe 555** and a foreign entry of **ann 555**. Because the Apollo entry is favored, **import_passwd** prompts you for a new UNIX ID for "ann."

Be aware, however, that Apollo reserved entries cannot be modified. (Table 4–2 list the Apollo reserved entries.) The **import_passwd** command will not modify a reserved entry even if it is the non–favored entry. For example, suppose you are running **import_passwd** with the foreign entry as the favored entry. During processing, it encounters a foreign group entry of **misc 0** and an Apollo group entry of **wheel 0**. Because group **wheel 0** is a reserved Apollo entry, you will be be prompted to modify the foreign entry, even though it is the favored entry.

Conflict Summary

Table 4–3 summarizes the effects of the identical user and favored entry options.

Table 4–3. Effects of Identical User and Favored Entry Options

Options Used	Foreign Entry	Apollo Entry	Result in Apollo Registry	Comments
–i, –a	joe 102	joe 555	joe 555	Name collision. Retain Apollo entry.
	joe 102	ann 102	ann 102	UNIX ID conflict. Request new UNIX ID for joe.
–i, –f	joe 102	joe 555	joe 102	Name collision. If 102 already exists in Apollo, prompt for new UNIX ID for that Apollo entry.
	joe 102	ann 102	joe 102	UNIX ID conflict. Request new UNIX ID for ann.
–a	joe 102	joe 555	joe 555	Name conflict. Request new name for joe 102, and if 102 is already defined in Apollo, a new UNIX ID as well.
	joe 102	ann 102	ann 102	UNIX ID conflict. Request new UNIX ID for joe.
–f	joe 102	joe 555	joe 102	Name conflict. Request new name for joe 555, and if 102 is already defined in Apollo, prompt for a new UNIX ID for that Apollo entry.
	joe 102	ann 102	joe 102	UNIX ID conflict. Request new UNIX ID for ann.

4.11.4 The import_passwd Syntax

This section presents the **import_passwd** command and describes the command arguments and options.

import_passwd [**–i**] [**–a** | **–f**] [**–c**] [**–o** *org*] **–s** *pathname* [**–v**]

–i	Identical name strings are not in conflict, but represent the same identity.
–a (D)	Favor Apollo entries.
–f	Favor foreign entries.
–c	Run in check mode: process the command showing conflicts, but make no changes to the files.
–o *org*	*org* is the name of the Apollo organization to be used for all imported account entries. The default is the organization named "none."
–s *pathname*	*pathname* is the path to the directory containing the foreign password and group files to be imported.
–v	Run in verbose mode: generate a verbose transcript of all activity.

4.11.5 Using import_passwd

This section describes conflicts you may encounter when running **import_password.**

Using Check Mode

You should first run **import_passwd** in check mode using the **–c** option. In this mode, **import_passwd** simulates the results of a processing run showing the conflicts that will be be encountered when **import_passwd** is run without the **–c** option.

Check mode gives you a good idea of the quantity and complexity of the potential conflicts. However, check mode does not make any changes to the file. When you run without the **–c** option and make changes to resolve conflicts, these changes can in turn create further conflicts not readily apparent in check mode.

If you encounter numerous conflicts in check mode, you may find it more efficient to manually edit either the registry or the UNIX group and password files to resolve some obvious conflicts before you run **import_passwd**.

Answering Prompts

When you run **import_passwd**, you may be prompted for names and numbers (UNIX IDs). The following conventions apply to your answers to these prompts:

- Names must
  - Begin with a lowercase letter
  - Contain *only* lowercase letters, digits, or underscore characters
  - Not exceed 32 characters in length, unless your system imposes UNIX restrictions, in which case, they cannot exceed 8 characters in length

  If your system imposes UNIX name restrictions, you should carefully evaluate the effect of these restrictions on existing entries in the files against which you are running **import_passwd**. Some of the entries may be longer that 8 characters.

- Numbers must range between 0 and 65535, inclusive. (We suggest that you not use numbers under 100, since these may be reserved in future releases.)

If you enter a name or number in an incorrect format, **import_passwd** will ignore your entry and re-prompt.

Processing Prerequisites

The Apollo registry must exist before you can use **import_passwd**. If you are simply adding a few Apollo nodes to a foreign network, you can create a new, but empty, registry to meet this requirement. Once the registry exists, the registry server must be running and you must be logged on as root.

Synchronizing Apollo UNIX IDs

If **import_passwd** makes any changes to Apollo UNIX IDs, you must run the **/etc/syncids** tool on every Apollo node in your network. This tool ensures that the changes are synchronized with the UNIX IDs stored on disk. On nodes that have more than one disk volume, you must run **syncids** on each volume.

If your Apollo registry is replicated before you run **syncids**, you should use the **lrep -state** command in the **/etc/rgy_admin** tool to verify that all slave registry servers are up to date. This command shows the time stamps of the most recent changes to each replica of the registry database. You should not run **syncids** until the time stamps for all slave servers match the time stamp for the master.

See the reference documentation for **syncids** and **rgy_admin** online or in the *Command Reference* manual for your environment.

Synchronizing Foreign UNIX IDs

If any of the conflicts found by **import_passwd** involve reserved information in the Apollo registry, you may have to resolve conflicts by changing names and IDs on your foreign systems. You also may have to change names and IDs on your foreign systems if you run **import_passwd** with the **-a** (favor Apollo) option. In that event, you must then use the **chown** and **chgrp** utility to update the UNIX IDs on all affected files and directories in the foreign file systems.

4.12 A Sample import_passwd Session

This subsection illustrates a simplified **import_passwd** session. The session uses the foreign group file and password files shown in Figure 4-3 and the Apollo group file and password entries shown in Figure 4-4.

Foreign Group File Entries

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
rje::8:rje,shqer
daemon::12:root,daemon
tgroup::35:
```

Foreign Password File Entries

```
root::0:1:0000-Admin(0000):/:
daemon::1:1:0000-Admin(0000):/:
bin::2:2:0000-Admin(0000):/bin:
sys::3:3:0000-Admin(0000):/usr/src:
adm::4:4:0000-Admin(0000):/usr/adm:
uucp::5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp::10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/
uucico
rje::18:18:0000-rje(0000):/usr/rje:
trouble::70:1:trouble(0000):/usr/lib/trouble:
lp::71:2:0000-lp(0000):/usr/spool/lp:
setup::0:0:general system administration:/usr/admin:/bin/rsh
powerdown::0:0:general system administration:/usr/admin:/bin/rsh
sysadm::0:0:general system administration:/usr/admin:/bin/rsh
checkfsys::0:0:check diskette file system:/usr/admin:/bin/rsh
makefsys::0:0:make diskette file system:/usr/admin:/bin/rsh
mountfsys::0:0:mount diskette file system:/usr/admin:/bin/rsh
umountfsys::0:0:unmount diskette file system:/usr/admin:/bin/rsh
```

Figure 4-3. Foreign Group and Password Entries

Group Entries

```
wheel::0:
daemon::1:
none::2:
backup::3:user
locksmith::4:
login::5:
mail::6:bin
bin::7:root
server::8:
sys::9:root
staff::10:
sys_admin::11:user
sys_proj::12:
tgroup::35:
```

Password Entries

```
root:sq1RclUrrb1L6:0:10::/:
daemon:sq1RclUrrb1L6:1:2::/:
none:sq1RclUrrb1L6:2:2::/:
user:sq1RclUrrb1L6:3:2::/:
lp:sq1RclUrrb1L6:4:7::/:
sys_person:sq1RclUrrb1L6:5:2::/:
admin:sq1RclUrrb1L6:6:2::/:
uucp:sq1RclUrrb1L6:7:2::/usr/spool/uucppublic:
bin:sq1RclUrrb1L6:8:7::/:
```

Figure 4–4. Apollo Group and Password Entries

The following subsections describe a session in which you perform the following tasks: invokethe command, examine the group entries, examine the password file, add members to groups, and update the registry.

Phase 1: Invoking import_passwd

1. In the sample session, the following form of the **import_passwd** command is entered at the shell prompt:

   ```
   $ import_passwd -s sys5.3_tapes/adm -i -v
   ```

   This command specifies the following:

   - Identical names represent the same identify (**–i**)
   - The pathname of the foreign file (**–s sys5.3_tapes/adm**)
   - Apollo entries should be favored (default of **–a**)
   - Not running in check mode (**–c** not specified)
   - Do run in verbose mode (**–v**)

2. The system puts the registry in maintenance mode, reads the foreign group and password files, and begins to create Apollo group entries in the Apollo registry:

   ```
   Preparing registry...
   Preparing foreign files...
   Creating Group entries from foreign group file...
   ```

Phase 2: Examining the Group Entries

If you examine Tables 4–2 and 4–3, you can see that the name **root** in the foreign and the name **wheel** in Apollo both have UNIX IDs of 0.

3. The **import_passwd** command tells you this and, since Apollo entries are favored, prompts for a new UNIX ID for the foreign entry:

   ```
   root::0:root
   ?(import_passwd) Group - UNIX id conflict
   Foreign Group: "root" 0  Apollo: "wheel" 0 (reserved).
   Please enter new UNIX id for Foreign Group 'root' 0: 100 <RETURN>
   >> Adding pgo entry for: root 100
   ```

   The value **100** is entered as the new UNIX ID for the foreign group entry. Note that **import_passwd** displays the foreign entry as it examines it (in the sample above, **root: :0:root**). And, because it is running in verbose mode, **import_passwd** describes the actions it is taking. Each such description is prefaced with the symbols >>.

   If you were running **import_passwd** in check mode, you would not be prompted for a UNIX ID. Instead, you would be informed of the need and processing would continue. The message would look like

   ```
   root::0:root
   ?(import_passwd) Group - UNIX id conflict
   Foreign Group: "root" 0  Apollo: "wheel" 0 (reserved).
       need new UNIX id for Foreign Group
         (UNIX id for Apollo "wheel" is currently 0)
   ```

4. The **import_passwd** command then finds another UNIX ID conflict:

   ```
   other::1:
   ?(import_passwd) Group - UNIX id conflict
   Foreign Group: "other" 1  Apollo: "daemon" 1 (reserved).
   Please enter new UNIX id for Foreign Group 'other'1: 101<RETURN>
   >> Adding pgo entry for: other 101
   ```

   Note that **import_passwd** tells you that the UNIX ID of **1** is reserved in Apollo. Because **1** is reserved, even if you had run **import_passwd** with the foreign entry favored, you would still be prompted for a new entry for the foreign UNIX ID. The **import_passwd** command will not change reserved entries.

5. If **import _passwd** finds no conflicts, it displays the group entries as it processes them and, because it is running in verbose mode, the actions it is taking:

```
bin::2:root,bin,daemon
>> Foreign Group: "bin" 2  Apollo: "bin" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id

    sys::3:root,bin,sys,adm
    >> Foreign Group: "sys" 3  Apollo: "sys" 9 (reserved)
    >>   -i specified - retaining Apollo UNIX id
```

   As it continues through the foreign group file, **import_passwd** finds two other UNIX ID conflicts: foreign entries **adm 4** and **rje 8** and Apollo entries **locksmith 4** and **server 8**, respectively.

Phase 3: Examining the Password File

6. **Import_passwd** then proceeds to create Apollo person and account entries from the foreign password file. It displays:

```
Creating Person entries and Accounts from foreign passwd file...
```

7. The first two entries are processed with no conflicts:

```
root::0:1:0000-Admin(0000):/
>> Foreign Person: "root" 0  Apollo: "root" 0 (reserved)
>>   -i specified and UNIX ids match - no change necessary
>> Adding account for root.other.none.

daemon::1:1:0000-Admin(0000):/
>> Foreign Person: "daemon" 1  Apollo: "daemon" 1 (reserved)
>>   -i specified and UNIX ids match - no change necessary
>> Adding account for daemon.other.none.
```

   The **import_passwd** command displays the names of the registry accounts it is creating. If you had changed a person or group name to resolve a previous name conflict, the new name would be displayed. For example, suppose that you had a foreign and an Apollo entry of **joe.other.none**. During **import_passwd** processing, you changed the name of the Apollo **joe** to **smith**. When **import_passwd** created the Apollo person and account entries, a verbose display would look like:

```
joe::1:1:0000-Admin(0000):/
>> Adding account for smith.other.none.
```

   The foreign entry is displayed first with its correct name, **joe**. The account being added to the Apollo registry is displayed by its new name, **smith**. **Import_passwd** keeps track of changes that are made and the relationships that existed before the changes were made.

8. The third entry produces the following warning message:

```
bin::2:2:0000-Admin(0000):/bin
    >> Foreign Person: "bin" 2  Apollo: "bin" 8 (reserved)
    >>   -i specified - retaining Apollo UNIX id
    >> Adding account for bin.bin.none .
    (WARNING) "bin.bin.none": Account already exists and is re-
    tained unchanged.
```

This message is notifying you that the **bin.bin.none** account exists in the Apollo registry. The Apollo account will retain its information (that is, abbreviation, account_valid, gecos, passwd, shell, home directory, passwd_dtm, and passwd_valid), even though the foreign account may have information that differs from the Apollo account.

9. Processing continues. The **import_passwd** command discovers two more UNIX ID conflicts (foreign **sys 3** and Apollo **user 3**; foreign **adm 4** and Apollo **lp 4**). Then **import_passwd** finds an un-named group in the foreign password file and a UNIX ID conflict.

   First **import_passwd** prompts you for a group name:

   ```
   uucp::5:5:0000-uucp(0000):/usr/lib/uucp
   >> Foreign Person: "uucp" 5  Apollo: "uucp" 7 (reserved)
   >>   -i specified - retaining Apollo UNIX id
   ?(import_passwd) Foreign Group 5 is unnamed.
       (UNIX id also conflicts with Apollo "login" (reserved) ).
   Please enter a name for Foreign Group 5: group_105 <ENTER>
   ```

   In the sample session, **group_105** is entered as the new name.

   Note that if you were running in check mode, **import_passwd** would supply a name for the group in order to keep processing. The prompt would look like:

   ```
   uucp::5:5:0000-uucp(0000):/usr/lib/uucp
   >> Foreign Person: "uucp" 5  Apollo: "uucp" 7 (reserved)
   >>   -i specified - retaining Apollo UNIX id
   ?(import_passwd) Foreign Group 5 is unnamed.
       (UNIX id also conflicts with Apollo "login" (reserved) ).
       Temporarily using the name of "group_5"
   ```

   Then, **import_passwd** asks you for the new UNIX ID for the group:

   ```
   ?(import_passwd) Group - UNIX id conflict
   Foreign Group: "g105" 5  Apollo: "login" 5 (reserved).
   Please enter new UNIX id for Foreign Group 'g105' 5: 105 <ENTER>
   >> Adding pgo entry for: g105 105
   >> Adding account for uucp.g105.none.
   ```

Phase 4: Adding Members to Groups

10. When **import_passwd** completes processing of the foreign password file, it adds the members to the groups:

    ```
    Adding memberships from foreign group file...
    root::0:root
      Member: root
    other::1:
    bin::2:root,bin,daemon
      Member: root
      Member: bin
      Member: daemon
    ```

    As it adds the members, **import_passwd** displays them. If there are no members added (as in group **other::1:**), none are displayed. If you are not running in verbose mode, only the names of the groups are displayed as they are processed, not the members being added.

11. When **import_passwd** processes the **rje::8:rje,shger** entry, it displays the following message:

```
rje::8:rje,shqer
  Member: rje
  Member: shqer
?(import_passwd) Cannot add member - Person is unknown or invalid
```

    In foreign UNIX systems, you can add members to groups even if the person has not been added as a person. In Apollo systems you cannot. The **import_passwd** command has found a member of a foreign group who has not been added as a person. Therefore, **import_passwd** will not add that member to the group.

Phase 5: Updating the Registry

12. Processing continues until **import_passwd** completes adding members to groups, it recaps the results of its processing:

```
0 name conflicts, 8 UNIX id conflicts, 3 unnamed groups,1 error,
2 warnings.
```

    And then asks if you want to save the changes:

```
Import operation successful.  Update registry [y/n]?  y <RETURN>
```

    In this sample session, the answer is Y (yes).

13. The **import_passwd** command then saves the changes and completes processing:

```
Closing registry...
Closing foreign files...
Done.
```

    Because the **-i** option was specified in the command used in the sample session, no name conflicts were encountered. A sample of the prompt resulting from a name conflict is shown below:

```
bin::2:root,bin,daemon
?(import_passwd) Group - Name conflict
Foreign Group: "bin" 2  Apollo: "bin" 7 (reserved)
Please enter new name for Foreign Group "bin": fbin <ENTER>
>> Adding pgo entry for: fbin 2
```

    In the sample above, a new name of **fbin** is assigned in the Apollo registry to entry representing the foreign group **bin**.

4.13 Local Registries

Each node can have a local registry that contains information about the node's most recent users and the date and time when they last logged in. If you log in to a node and no registry server is available on the network, the operating system checks the node's local registry for information about your account. If the information exists in the local registry, the system allows you to log in with that account; if not, it refuses to log you in. If both the network registry and the local registry are unavailable, the system will always allow you to log in with the account name **user.none.none**.

4.13.1 Setting Up the Local Registry

The operating system creates a local registry the first time a user logs in to a node, provided a registry server is running somewhere on the network. Thereafter, the local registry is updated every time someone logs in to the node. Users can edit some of the information in a node's local registry by invoking the **edrgy –l** command on that node.

4.13.2 Running a Small Network without Registry Servers

If you have a small network consisting of one or two nodes with no more than 10 accounts, it is possible to operate your network by using only local registries. You need to run **rgyd** while you create accounts, then copy account information into the local registry on each node, as follows:

1. Start a **rgyd** process on one node and use **edrgy** to create the accounts you need.
2. Use the **copy** command in **edrgy** to add account information to the local registry. The command **copy %.%.%** will fill the local registry to capacity.
3. Use the **stop** command in **rgy_admin** to terminate the **rgyd** process.

The accounts you created now have access to the nodes via the local registry. Of course, if you wish to change any account information, you must restart **rgyd** to make the changes.

(Even without local registries, all reserved entries are available to the operating system, so users on your network can log in as **user.none.none**.)

Chapter 5

Protection of Files and Directories

Contents

Chapter 5

Protection of Files and Directories

This chapter describes how SR10 Domain/OS controls access to files and directories. Domain/OS implements the UNIX protection model and extends it with an Access Control List (ACL) model. Under Domain/OS, UNIX protection operates as you would expect on other UNIX systems, while the ACL mechanism provides greater flexibility in controlling access to objects.

Section 5.1 describes UNIX protection. Section 5.2 describes the ACL model and the ways in which it extends the UNIX model. Section 5.3 explains how Domain/OS derives UNIX modes from ACLs. (If you use only vanilla UNIX protection, without any of the ACL extensions, you can skip Sections 5.2 and 5.3.) The remaining sections discuss protection in the context of system administration.

NOTE: The SR10 protection–checking mechanism can operate over the network with the mechanism released at SR9.7, but not with those released prior to SR9.7. Therefore, if your site is upgrading to SR10 over a period of time and you want to share files between SR10 and non–SR10 nodes, you should ensure that all non–SR10 nodes have system software of SR9.7 or later vintage. For information about transition to SR10 and about operating an environment with SR9.7 and SR10 nodes, see *Making the Transition to SR10 Operating System Releases.* For information about installing software, see *Installing Software with Apollo's Release and Installation Tools.*

5.1 UNIX Protection

The UNIX protection scheme is based on owner and group IDs. Every file system object has associated with it an owner ID and a group ID. Every process has four IDs: a real owner ID, an effective owner ID, a real group ID, and an effective group ID. Real IDs always represent the actual owner and group of the process; effective IDs are used by the system for checking rights.

5.1.1 Protection Modes

Each file and directory has permissions for three categories of users: owner, group, and others. These permissions are represented as a UNIX **mode**, an octal number constructed from the logical OR of the following bits:

4000	set user ID on execution
2000	set group ID on execution
1000	sticky bit
0400	read by owner
0200	write by owner
0100	execute by owner
0040	read by group
0020	write by group
0010	execute by group
0004	read by others
0002	write by others
0001	execute by others

For example, mode 640 allows the owner to read and write, the group to read only, and others no access at all.

The UNIX command **ls**, when given the **-l** option, reports an object's octal mode in the form

```
-rwxrwxrwx
```

where each rwx sequence stands for the set of permissions available to the owner, the group, and others, respectively. (The initial hyphen indicates that the object is an ordinary file. For a directory, the initial character is the letter "d"; for a symbolic link, it is the letter "l".) Mode 640, for example, appears as

```
-rw-r-----
```

Table 5-1 shows these rights and their meanings for files and directories.

Table 5-1. UNIX Permissions

	File	**Directory**
r	Read	List entries
w	Write	Add, delete, or change entries
x	Execute	Search

5.1.2 The setuid and setgid Bits

The 4000, 2000, and 1000 bits in a protection mode have meaning only for executable files. They affect the behavior of a file when a user executes it.

The 4000 bit is the **setuid** bit. If you execute a file that has this bit on, the operating system sets the effective user ID of the process to the owner ID of the file. Similarly, the 2000 bit is the **setgid** bit. It sets the effective group ID to the group ID of the file.

The **ls** command reports the setuid (or setgid) bit with the letter "s" in place of the "x" for execution by owner (or group). Though **ls** displays them in place of the "x," these bits do not affect permission to execute the file. The operating system always checks the "x" bit.

For example, if a program has the protection mode 6711, the owner ID **pooh**, and the group ID **bears**, **ls –l** reports its protection as

```
    -rws--s--x
```

The program always executes as though it were invoked by user **pooh** and group **bears**.

The 1000 bit is the **sticky** bit. On Domain systems, this bit has no effect.

5.1.3 Checking Permissions

When a process attempts to access a file system object, the operating system checks permissions in order: first owner, then group, and finally others. The first matching permissions apply. Permissions for the owner apply to any process whose effective user ID matches the object's owner ID. Group permissions apply to any process whose effective group ID matches the object's group ID and whose user ID does not match the object's owner ID. Permissions for others apply to all other processes.

For example, suppose the file **tarts** has the owner ID **jack**, the group ID **hearts**, and the protection mode 640. With the **–l** and **–g** options, **ls** will show complete protection information for **tarts**:

```
$ ls -lg tarts
-rw-r-----   1 jack     hearts       3474 Feb 29 10:54 tarts
```

Processes with an effective user ID of **jack** will have both read and write permissions for **tarts**. Other processes will have read permission if their effective group ID is **hearts** and otherwise will have no access at all.

The **/etc/passwd** file specifies a default group for each user. Additional groups can be specified in the **/etc/group** file. Under SysV, users can invoke the **newgrp** command to change their effective group IDs. Under BSD, a user effectively has membership in several groups at once: the default group specified in the **passwd** file and any others specified in the **group** file.

In Domain/OS, the PROJLIST environment variable determines which scheme for group membership applies. PROJLIST is set by default in the BSD operating environment. In other operating environments you can set it optionally. (See Chapter 4, Subsection 4.2.4 for details.)

Because the operating system always applies the first matching permissions, it is possible for owner of a file to have fewer permissions than other users on the system. Mode 477, for instance, gives the owner read rights only, but gives everyone else all rights.

5.1.4 Assigning and Changing Permissions

A newly created object inherits its owner and group IDs from the process that creates it. (Except that under 4.3BSD, an object inherits the group ID from the parent directory, the directory where the object is created.)

The permissions for a newly created object can be specified by the creating process, through the *mode* argument to system calls such as **mkdir** or **open**. System calls that do not take a *mode* argument create objects with mode 777. In both cases, the permissions specified by the creating call are then filtered through the **umask** of the process. The umask is a bitmask that specifies permissions to be disallowed for any objects created by the process. For example, if a process with a umask of 022 creates an object via an **open** call with a *mode* of 770, the object will have a permissions mode of 750.

Only an object's owner or the super–user can change the IDs and permissions associated with the object. (The super–user, which we discuss in Subsection 5.4.1, has rights outside the normal protection model.) On vanilla 4.3BSD systems, only the super–user can change the owner ID of an object. However, Domain/OS BSD does not implement this restriction.

5.1.5 Utilities

This subsection surveys the UNIX utilities for inspecting and modifying protection. See the *BSD Command Reference* and the *SysV Command Reference* for detailed descriptions.

The **umask** command (built in to the Aegis shell as well as the UNIX shells) sets or displays the value of the umask. The **chmod** command changes the permissions mode of a file or directory; **chown** changes the owner ID, and **chgrp** command changes the group ID. To display the protection of an object, you can use **ls** with the **–l** and **–g** options.

Figure 5–1 illustrates the use of these utilities. In this example, files are created via the **touch** command, which uses the **creat** system call with a *mode* of 666.

```
$ umask
0
$ touch magritte
$ ls -lg magritte
-rw-rw-rw-   1 mk       none          0 Feb  5 14:48 magritte
$ umask 022
$ touch dali
$ ls -lg dali
-rw-r--r--   1 mk       none          0 Feb  5 14:49 dali
$ chmod 664 dali
$ chgrp dds dali
$ ls -lg dali
-rw-rw-r--   1 mk       dds           0 Feb  5 14:49 dali
```

Figure 5-1. UNIX Protection Utilities

The *BSD Programmer's Reference* and the *SysV Programmer's Reference* describe system calls pertaining to protection, including **stat**, **chmod**, **chown**, and **umask**.

5.2 Domain/OS Protection

The SR10 Domain/OS protection model incorporates UNIX semantics and extends them with Access Control Lists (ACLs). The Domain/OS model is thus a superset of the UNIX model, as shown in Figure 5-2.

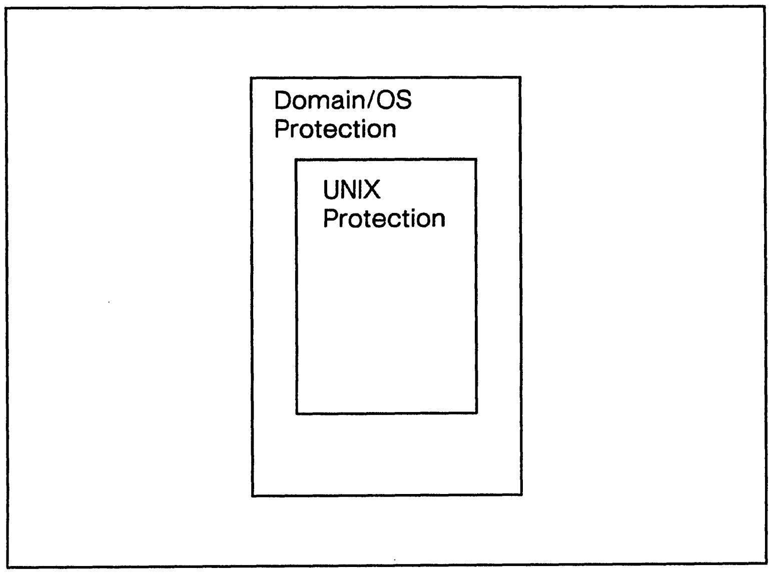


Figure 5-2. Domain/OS and UNIX Protection

5.2.1 ACL Structure

Each file system object has an ACL that defines who can perform what operations on that file or directory. An ACL contains a number of entries. Each ACL entry has two parts: a **Subject Identifier** (SID) and a set of **access rights.**

Subject Identifiers

An SID is a specifier consisting of fields for a person, a group, and an organization. In ACL entries, each field of the SID can be either a name or a wildcard (%). Section 4.2 describes SIDs in more detail.

Access Rights

Access rights describe the extent of an SID's access to the associated object. Table 5-2 lists the access rights for files and directories.

Table 5-2. Domain/OS Access Rights

Access	**File**	**Directory**
r	Read	List entries
w	Write	Add, delete, or change entries (including links)
x	Execute	Search
p	Change protection	Change protection
k	Keep (prevents deletion or change of name)	Keep (prevents deletion or change of name)

The **r**, **w**, and **x** rights are like their UNIX counterparts.

In vanilla UNIX systems, only the owner of an object or the super-user can change the object's protection. However, in Domain/OS, you can use the **p** access right to grant any SID permission to change the object's protection.

The **k** right in an ACL entry denies the specified SID the right to delete the object or to change its name, even if that SID has write permission for the parent directory. Under traditional UNIX semantics, all objects in a writable directory are deletable. However, in some situations, you want a directory to be writable, but you want to protect a particular file or subdirectory from deletion. You can establish this protection by setting the **k** right in the ACL for the file or subdirectory.

If the ACL entry for an object contains both **p** and **k** rights, the specified SID can delete the object by using the **-f** option to either of the Aegis commands **dlf** and **dlt**. However, the UNIX command **rm** will not delete the object unless invoked by the super-user.

5.2.2 Types of ACL Entries

An ACL must contain four **required entries**; in addition, it can contain **extended entries**. The required entries, which express UNIX protection information, are stored in the i-node of each file or directory to provide rapid access for operations like **stat** and **chmod**.

Required Entries

Every ACL contains at least four entries: one each for person, group, organization, and world. These are the four required entries in an ACL. The person, group, and world entries correspond to the UNIX concepts of owner, group, and other. The organization entry does not have a UNIX analog.

A required ACL entry can be designated as an **ignored entry**. When the operating system checks permission to access the object, it ignores the entry. The **acl** command displays the entry with the word "ignored" in place of the access rights.

The SIDs in the four required entries must be of the forms *person.%.%*, *%.group.%*, *%.%.organization*, and *%.%.%*.

Extended ACL Entries

In addition to the four that are required, an ACL can contain up to 22 extended entries. As we will explain in Subsection 5.3.2, the operating system applies a mask to the rights specified in extended entries when it checks permissions.

The SIDs in extended entries can specify any combination of *person*, *group*, and *organization*, provided they do not duplicate any SIDs in the required entries. An extended entry cannot be ignored.

An Example

Figure 5-3 shows the ACL for a file as reported by the **acl** command. The left column shows SIDs; the right column shows the access rights associated with each SID.

```
$ acl bar
Acl for bar:
Required entries
 mk.%.%                          prw--
 %.dds.%                         -rw--
 %.%.r_d                         [ignored]
 %.%.%                           -r---
 Extended entry rights mask:     -r---
Extended entries
  %.backup.%                      -r---
  curly.stooges.%                 -----
```

Figure 5-3. An ACL

5.2.3 Types of ACLs

There are four types of ACL: the file ACL, the directory ACL, the initial default file ACL, and the initial default directory ACL. The first type is associated with files; the other three are associated with directories. ACLs of all four types must contain the required entries for owner, group, organization, and world.

The **file ACL** controls access to the file with which it is associated.

The **directory ACL** controls access to the directory with which it is associated.

The **initial file ACL** determines the protection for files created within a directory.

The **initial directory ACL** determines the protection for subdirectories created within a directory.

Initial ACLs determine the protection assigned to newly created files and directories. Subsection 5.2.5 discusses in detail the role of initial ACLs.

5.2.4 How ACLs are Interpreted

The operating system associates each process with an SID. When a process attempts to access a file or directory, the operating system compares the process' SID with the entries in the ACL for the file or directory; it applies the access rights specified in the first entry whose SID matches the process' SID. If the rights allow the requested mode of access, the process gains access; if not, access is denied.

Because the first matching ACL entry determines a process' access to an object, the order of ACL entries is important. In general, ACL entries are ordered from the most specific to the least specific, with special precedence for required entries. The person field is more specific than the group field, which in turn is more specific than the organization field.

Figure 5–4 illustrates the rules for ordering of ACL entries.

mk.%.%	prwx–	(required)
mk.dds.r_d	prwx–	
mk.dds.%	prwx–	
mk.%.r_d	prwx–	
user.unix.%	–r–––	
user.%.%	–r–––	
%.unix.%	–rwx–	(required)
%.dds.r_d	–rwx–	
%.dds.%	–rwx–	
%.%.r_d	–r–x–	(required)
%.%.legal	–r–––	
%.%.%	–––––	(required)

Figure 5–4. Effective Order of Entries in an ACL

Note that Figure 5–4 shows the effective order of ACL entries but does not correspond to the output of the **acl** command; **acl** displays extended entries separately from required entries. Figure 5–5 shows the same ACL, as **acl** would display it.

```
$ acl foo
Acl for foo:
Required entries
 mk.%.%                          prwx-
 %.unix.%                        -rwx-
 %.%.r_d                         -r-x-
 %.%.%                           -----
 Extended entry rights mask:     prwx-
Extended entries
  mk.dds.r_d                      prwx-
  mk.dds.%                        prwx-
  mk.%.r_d                        prwx-
  user.unix.%                     -r---
  user.%.%                        -r---
  %.dds.r_d                       -rwx-
  %.dds.%                         -rwx-
  %.%.legal                       -r---
```

Figure 5–5. An ACL as Displayed by the acl Command

5.2.5 How ACLs are Assigned to New Files and Directories

When a process creates a file or directory, the operating system assigns an ACL to the new object. Three factors govern the assignment of ACLs:

- The protection mode, if any, specified by the program creating the object
- The **umask**, if any, of the process creating the object
- The initial ACLs of the **parent directory**, the directory in which the object is created

Creation Mode and the umask

Some UNIX system calls, including **open** and **mkdir**, take a *mode* argument that allows you to specify a protection mode for a newly created object. Some other UNIX system calls and all Aegis system calls specify the widest possible permissions, allowing all forms of access for all users. In either case, this **creation mode** is filtered through the umask of the process, as Subsection 5.1.4. explains.

In Domain/OS, the low-order octal digit in the *mode* and the umask applies both to the required organization entry and to the %.%.% entry. The middle digit applies to the required group entry. The high-order digit applies to the required person entry.

The permissions that result from application of the umask to the creation mode are either assigned to the new object or overridden by entries in the initial file or directory ACL of the parent directory. We refer to the first mechanism as "inheritance from the process" and the second as "inheritance from the parent directory." The choice between these mechanisms is specified within the initial ACL itself.

Initial ACLs

Domain/OS allows the SID and/or the rights of a required ACL entry to be inherited either from the creating process or from the parent directory. Inheritance is specified as part of the initial ACLs of the parent directory and can apply to the SID, to the rights, or to both parts of an entry.

Figure 5-6 shows an initial file ACL that implements BSD protection semantics. The rights for the required organization entry are ignored. All other SIDs and rights are inherited from the creating process except for the group SID, which derives from the parent directory.

```
$ acl -if figaro
Initial (default) acl for files created under figaro
Required entries                                       For the current process:
 [from process]              [specified by process]    mk.%.%
 %.dds.%                     [specified by process]
 %.%.r_d                     [ignored]
 %.%.%                       [specified by process]
 Extended entry rights mask:  -----
```

Figure 5-6. Initial File ACL Implementing BSD Inheritance

As Figure 5-6 shows, **acl** reports under the heading "For the current process" the particular SIDs that will be inherited if the current process creates an object.

The **edacl** command has options that set inheritance of ownership or permissions in the initial ACLs of a directory. However, you should not ordinarily need to use these options; when Domain/OS software is installed on your node, the initial ACLs of your directories are set to implement the inheritance mechanism appropriate for your environment. The **chacl** command allows you to change easily the inheritance semantics for any particular directory.

5.2.6 Utilities

This subsection surveys the utilities that deal with ACLs.

The Aegis operating environment provides the utilities **acl** and **edacl**. You can use **acl** to display an ACL or to copy an ACL from one object to another. You can use **edacl** to edit ACLs.

In all three of the operating environments, the utilities **chacl**, **lsacl**, and **cpacl** are available in **/usr/apollo/bin**. These tools offer functionality similar to that of the Aegis **acl** and **edacl** utilities but user interfaces similar to those of the UNIX **chmod**, **ls**, and **cp** utilities.

If you use BSD or SysV UNIX protection without any of the Domain/OS extensions, you will not need to deal with ACLs at all. See Subsection 5.1.5 for a survey of the UNIX protection utilities.

5.3 How UNIX Modes are Derived from ACLs

This section describes how Domain/OS protection is translated into UNIX protection.

Domain/OS derives the modes that it supplies to UNIX programs and calls from the access rights in the ACL. If the protection of an object involves Domain/OS extensions to the UNIX protection model, the ACL information might not map directly to UNIX modes. In such cases, the operating system will over-represent rather than under-represent the access available when it reports permissions, and it will over-restrict rather than under-restrict access when it sets permissions.

NOTE: The information in this section is important only for those who use the Domain/OS extensions to UNIX protection. Vanilla UNIX protection behaves as you would expect on other UNIX systems.

5.3.1 Permissions for Owner and Group

As Figure 5-7 shows, permissions for the owner and group of an object are those specified in the required ACL entries for the person and group.

```
$ acl foo
Acl for foo:
Required entries
 mk.%.%                            prwx-
 %.dds.%                           prw--
 %.%.r_d                           -rw--
 %.%.%                             -----
 Extended entry rights mask:       -rwx-
Extended entries
  user.%.%                          ---x-
  %.osdev.%                         -rw--
  %.backup.%                        -r---
$ /bin/ls -lg foo
-rwxrw-rwx  1 mk       dds           0 Nov 10 17:35 foo
```

Figure 5–7. ACL Entries and UNIX Permissions for a File

5.3.2 Permissions for Others

Since ACLs do not provide a direct equivalent to UNIX "other" rights, the operating system derives these rights from information in the ACL. The ACL entries that determine UNIX "other" rights are the required organization entry, the required world entry, and any extended entries—that is, all except the required person and group entries.

To expedite the mapping of ACL information to UNIX rights, the operating system maintains for every file or directory a set of rights that summarizes the permissions allowed by extended entries. These rights are reported in the output of **acl** as the "extended entry rights mask" and in the output of **lsacl -l** as the "extended entry mask." Unless otherwise set by a **chmod** call or command, the mask is the logical OR of all rights in any extended entries.

The remainder of this subsection discusses how the rights mask, the org rights, and the world rights are used when the operating system reports, checks, or sets UNIX rights.

Reporting Permissions

When UNIX "other" rights are required by a command such as **ls** or a call such as **stat**(2), the operating system computes and reports the logical OR of the following:

- The rights in the required *org* entry
- The rights in the required *world* entry
- The extended entry rights mask

In Figure 5–7, for example, **ls** reports "other" rights as **rwx** because each of these rights is available to some SID other than the owner and the group, even though none of those SIDs has all of the rights.

Setting Permissions

The rights mask reflects changes you make via **edacl** or **chacl** to the extended entries in an object's ACL. You can set the mask itself, independently of any ACL entries, via **chmod**.

If you use **edacl** or **chacl** to edit extended entries in the object's ACL, the operating system recomputes the rights mask by taking the OR of all rights in the extended entries. The mask does not change if you edit only required entries.

If you use the **chmod** call or command to change the rights for "others" on the object, the operating system implements the change as follows:

- It sets the rights mask to match the rights you specified for "others."
- It sets "world" (%.%.%) rights to those you specified for "others."
- It sets the required *org* entry to be ignored.
- It does not change any extended entries.

As a result, no SID can have any rights not allowed in the **chmod** call or command.

Figure 5–8 illustrates the use of **chmod** to change UNIX group and "other" rights on a file.

Checking Permissions

When it checks permissions for a extended entry, the system takes the logical AND of the rights mask and the permissions in the entry. Thus, SIDs in extended entries are allowed at most the access permitted by the rights mask.

After the **chmod** command in Figure 5–8, for example, **%.osdev.%** and **%.backup.%** can only read **foo**, and **user.%.%** can only execute it. The **w** in the entry for **%.osdev.%** is masked.

```
$ acl foo
Acl for foo:
Required entries
 mk.%.%                              prwx-
 %.dds.%                             prw--
 %.%.r_d                             -rw--
 %.%.%                               -----
 Extended entry rights mask:         -rwx-
Extended entries
  user.%.%                            ---x-
  %.osdev.%                           -rw--
  %.backup.%                          -r---
$ /bin/ls -lg foo
-rwxrw-rwx  1 mk       dds             0 Nov 10 17:35 foo
$ /bin/chmod 775 foo
$ acl foo
Acl for foo:
Required entries
 mk.%.%                              prwx-
 %.dds.%                             -rwx-
 %.%.r_d                             [ignored]
 %.%.%                               -r-x-
 Extended entry rights mask:         -r-x-
Extended entries
  user.%.%                            ---x-
  %.osdev.%                           -rw--
  %.backup.%                          -r---
$ /bin/ls -lg foo
-rwxrwxr-x  1 mk       dds             0 Nov 10 17:35 foo
```

Figure 5-8. Use of chmod to Set Permissions

5.4 Special Groups and Accounts

This section discusses special groups and accounts that you can use in managing Domain systems. These groups and accounts are **reserved**: they are added automatically and permanently to the registry database when the database is initialized. Subsection 4.2.3 gives a complete list of reserved names and accounts; here we discuss only those which pertain directly to the protection of system software.

5.4.1 The root Person and the locksmith Group

The operating system provides a special super–user named **root**. The root person is the most powerful on the system. It overrides all protections in the ACL mechanism. The root person always has the UNIX user ID 0.

Use of root accounts should be carefully controlled. We suggest strongly that you assign passwords to root accounts, restrict knowledge of the passwords, and change the passwords periodically.

To use a root account, you can either log in as a root user or invoke the UNIX **su** command to switch your user ID to root.

Domain/OS also provides a super–user group named **locksmith**. Accounts of the form **%.locksmith.%** have the same overriding privileges as those of the form **root.%.%**. Of course, locksmith accounts warrant the same careful control. The locksmith group is not allowed to appear on project lists, so members of this group do not carry super–user privileges unless they explicitly log in with locksmith accounts.

5.4.2 The sys_admin, staff, and wheel Groups

Any operating system includes software whose use should be restricted. In Domain/OS, access to system software (both programs and data files) is restricted to persons such as **root** and **bin** or groups such as **sys_admin**, **staff**, or **wheel**. To protect your system software from corruption or deletion, you should create accounts with these persons and groups only for users who need to have access to system software.

5.4.3 The server Group

The **/etc/server** shell command and the **cps** Display Manager command create server processes and assign to them the SID **user.server.none**. Processes started via **server** or **cps** run independently of log–in activity.

When you log out, the DM does not terminate any process that run with server group identity. Therefore, we recommend that you do not create any accounts with the **server** group.

5.4.4 The user.none.none SID

The registry database contains a reserved account with the SID **user.none.none**. In the event that no registry servers are available, the operating system will allow anyone to log in under this SID without giving a password. This is the only SID under which you can log in when the registry is unavailable. Of course, when a registry server is available, the password is required.

You can use ACLs to limit the rights available to this SID.

5.5 Backups

Any user performing a backup must have the following access rights:

- Read access to all files and directories being backed up
- Write access to the **backup_history** file in the directory at the top of any tree to be backed up
- Execute (that is, search) access to all directories in the pathnames of objects being backed up

You can meet these requirements simply by performing all backups as **root**. If you prefer not to give root accounts to everyone who performs backups, we suggest the procedure below, which enables any **%.backup** account to run the backup utility with root privileges. The procedure creates a copy of **wbak** that runs with its user ID set to root and can be executed only by members of the group **%.backup**.

1. Use the **edrgy** command to create accounts with the group name **backup**.
2. Make a copy of the **wbak** utility. Set its protection to be setuid, owned by the **root** person and the **backup** group, and executable only by **root** and **%.backup**. (You must use a **root** or **%.locksmith** account to set the protection.)

```
$ cd /usr/apollo/bin          (UNIX environments)
$ cp wbak wbak.priv
$ chown root wbak.priv
$ chgrp backup wbak.priv
$ chmod 4550 wbak.priv

$ wd /usr/apollo/bin          (Aegis)
$ cpf wbak wbak.priv
$ edacl -p root prx -g backup rx -o none -ignore -w -none -setuid
  wbak.priv
```

You should perform Step 2 on every node that will run backups. Now the **wbak.priv** copies, executable only by **root** and **%.backup** accounts, will run as **root**. The original copies of **wbak** remain executable by any user, but without root privileges.

5.6 Protected Subsystems

A protected subsystem associates a set of data files (subsystem **data objects**) with programs (subsystem **managers**) that have special access rights to those files. *Using Your Aegis Environment* describes how to create protected subsystems. In this section, we discuss the login protected subsystem.

The following list shows the shell commands that are part of the login subsystem.

- The **login** command allows users to change their SIDs while logged in.
- The **/sys/siologin/siologin** command allows users to log in to a Domain node via a terminal or modem attached to a serial I/O (SIO) line.
- The **/sys/spm/spmlogin** command allows users to log in to one Domain node from another via the **crp –on** command.

Refer to the *Aegis Command Reference* for information about the shell commands listed above.

A user must "enter" a subsystem in order to set its attributes. Permission to enter a subsystem requires read and execute rights on the file /sys/subsys/*subsystem*. Figure 5–9 shows the ACL for the **login** subsystem.

```
$ acl /sys/subsys/login
Acl for /sys/subsys/login:
Subsystem login manager
Required entries
 root.%.%                          pr-x--
 %.sys_admin.%                     pr-x--
 %.%.none                          -r----
 %.%.%                             -r----
 Non-required entry rights mask:   ------
```

Figure 5–9. ACL for the login Subsystem

5.7 Control of Remote Access

The **lprotect** command and the **–lao** option to **edacl** allow you to control access from remote machines to objects on your machine.

The **node_owners** file in **'node_data** determines who can run **lprotect** on a node and also controls rights to perform other operations such as killing processes.

5.8 Installation and Protection

The **invol** system utility sets protection on some system software. Other system software is protected as it is installed. You should not alter this protection because the operating system depends on it. For example, some UNIX subsystems require that programs, data files, and directories be owned by particular users and groups or assigned particular permissions.

Note that protection model used and implemented by SR10 installation procedures depends on inheritance and the type of inheritance depends on how the disk was **invol'd**. You should be aware that, following installation, certain directories are not protected by inheritance, including **'node_data** and **/systest**.

Refer to *Installing Software with Apollo's Release and Installation Tools* for a description of how protection is inherited during installation. Refer to the BSD and SysV Command References for details on **invol**.

5.9 Registries and Protection

The registry database is protected in ways that do not use ACLs directly. Every entry in the registry database has an **owner** with rights to modify information pertaining to the entry. Owners of registry information are specified by SIDs; these can include wildcard (%) fields. See Chapter 4 for complete information about the registry.

Chapter 6

Administrative Commands

Contents

NOTE: This chapter contains manual pages for selected TCP/IP commands. Refer to *Configuring and Managing TCP/IP* for a description of all TCP/IP commands.

NAME

cpboot – copy the system boot file sysboot

SYNOPSIS

/etc/**cpboot** *source_directory* *target_directory*

DESCRIPTION

cpboot copies the system boot file **sysboot** from one directory to another. The **sysboot** file is used by the bootstrap prom to start the system. **cpboot** is useful for copying **sysboot** to a floppy disk, thus making the standalone utilities (**sau**) directory on the floppy disk accessible from the boot prom. You may also use it to update a Winchester disk when a new software release is distributed.

source_directory (required)	Specify directory containing the file **sysboot**.
target_directory (required)	Specify directory to which **sysboot** is to be copied. This must be the entry directory on the target logical volume.

NAME

ctnode – catalog a node in the network

SYNOPSIS

/etc/ctnode [*options*] [*node_name* [*net.*]*node_id* ...]

DESCRIPTION

ctnode informs the local node that a remote node exists, thereby enabling network file access to the remote node. The command catalogs the *node_name* in the local copy of the network root directory as the entry directory for the remote node. In other words, **ctnode** adds the directory *//node_name* to your copy of the network root directory.

For information on deleting a *node_name* entry, type **help uctnode.**

We assign a node ID to every node during the manufacturing process. To find out the node ID of a node, type the following command at its keyboard:

```
$ lcnode –me
```

from another node's network root into your own, or any other node's network root. The merge options (**–md** and **–ms**) add the entry for a node to the target, provided the entry does not already exist and the source has exactly one entry for that node. In the case of one source and one target entry that match for a node, those entries are assumed to be correct. All other cases are considered to be ambiguous and the "confusion-resolution protocol" is invoked.

This "confusion-resolution protocol" first attempts to verify the correct entry name with the node itself. If the node is available, the reply from the node is cataloged regardless of whether **–md** or **–ms** is used because an answer from the node itself is assumed to be the truth.

If the node is unavailable to resolve an ambiguity, the entry containing the most recent UID (latest time stamp portion of the UID) is used. In this case, existing entries in the target directory are only updated if the **–ms** option is used. Multiple name/ID pairs are permitted.

If you do not specify **–n**, **–update**, or **–from**, the *node_name* and *net.node_id* arguments are required.

node_name (optional) Specify the name of the node you wish to catalog. If the *net.node_id* argument is specified, then *node_name* is required.

Default if omitted: you must use **–n, –update**, or **–from**

[*net.*]*node_id* (optional)

Specify the hexadecimal ID (and optional network ID) of the node you wish to catalog. The node must be connected to the network when this command is executed. If the *node_name* argument is specified, then *node_id* or [*net*].*node_id* is required.

Default if omitted: you must use **–n, –update**, or **–from**

Multiple name/ID pairs are permitted.

OPTIONS

If you do not specify **–n, –update**, or **–from**, the *node_name* and [*net.*]*node_id* arguments are required. The **–n, –update**, and **merge** options work only for remote nodes running Aegis SR5.0 or later. The [*net.*]*node_id* forms work only when both the local and remote nodes run Aegis SR9.0 or later.

–root Catalog *node_name* as the entry directory name for *node_id* in both the master network root directory and the local copy of the network root directory. This option is valid only if the *node_name* and *node_id* arguments are specified. This option is not valid if the **–n** option is specified.

–n [*net.*]*node_id...* Copy the entry directory name from the network root directory of the specified remote node to the network root directory of the local node. You do not need to know the entry directory name. However, you must specify the *node_id* or the *net.node_id* of the remote node. Multiple *node_id*'s and *net.node_id*'s may be specified. Use this option instead of the *node_name* net.*node_id* argument pair. This option is not valid if the **–r** option is specified.

–update Obtain a list of nodes currently responding to a network inquiry and perform the same operation as **–n** for each node. Names are replaced with the most current version, if they already exist in your local copy of the network root directory, and new names are added.

–from *//node ...*	Look in the specified list of network root directories for the names to add to the target network root, or use this network root as the source for names to merge into the target network root. Wildcards may be used to specify source node names. The **–from** option is not supported in a Domain internet environment.
–md	This option is used with **–from**. Merges all names in the source network root into the target network root. Preference is given to existing names in the target if there are unresolved conflicts (see the discussion of "confusion-resolution protocol" above).
–ms	This option is the same as **–md**, except that preference is given to entries in the source network root when there are unresolved conflicts (see the discussion of "confusion-resolution protocol" above).
–on *//node ...*	Catalog names in the network root of the specified nodes instead of the local network root. Wildcards may be used to specify target node names. The **–on** option is not supported in a Domain internet environment.
–r	Replace cataloged names if they already exist. An error occurs if you do not specify this option and try to add a *node_name* that has already been cataloged (unless you are using **–update**).
–l	List node names as they are cataloged.
–idupl	Ignore entry (suppress error messages) if name already exists in the target.
–lc	List invocations and resolutions of the "confusion-resolution protocol".

EXAMPLES

Add the node whose ID is 21 and whose entry directory name is os to your node's catalog:

```
$ /etc/ctnode os 21
```

Bring your node's catalog up to date with any new nodes on the network:

```
$ /etc/ctnode –update
```

Copy names os and **eve** from the network root on //**master**.

```
$ /etc/ctnode os eve –from //master
```

Add node ID 21 with the name os to the network root of all nodes whose names begin with "a".

```
$ /etc/ctnode os 21 –on //a?*
```

Merge network root of os into local network root, resolving conflicts:

```
$ /etc/ctnode –md –from //os
```

NAME

dtcb – dump contents of tcp control blocks

SYNOPSIS

/etc/dtcb [–f] [[–t]
  [<*tcb addr*>|**–a**] |**–u**
  [<*ucb_addr*> | **–a**]]

DESCRIPTION

The command **dtcb** dumps the contents of the **tcp** control blocks associated with a particular tcp connection. The address of the **tcb** to be dumped may be obtained using the **netstat** program. Two control blocks are dumped: the **ucb** (user control block) which contains the send and receive queues and user-related flags, and the **tcb** (tcp control block) which contains the connection sequence numbers, state, flags, and out-of-sequence queues.

OPTIONS

–f	Force output if tcpd not running.
–t <*tcb_addr*>	Hexadecimal address of a **tcb** or, if not supplied, all **tcb**s.
–u <*ucb_addr*>	Hexadecimal address of a **ucb** or, if not supplied, all **ucb**s.
–a	All (both **tcb**'s and **ucb**'s for each socket).

EXAMPLES

The dump of a tcp control block for a listening ftp connection might look like this:

```
$ /etc/dtcb -t 1A9A50
ucb at 0x1A99C4:
local 0.0.0.0  lport 21
host  0.0.0.0  fport 0
uc_snd 8192  uc_ssize 0  uc_rcv 8192  uc_rsize 0
uc_shead 0  uc_stail 0  uc_rhead 0  uc_rtail 0
xflag:
UREUSEADDR UCANACCEPT
iostate:

status:
UCLOSED
flags:
UTCP
oobmark 0  oobcnt 0
```

```
TCB at 0x1A9A50:
lport 0x0  fport 0x0
t_state LISTEN
irs 00000000 rcv_urp 00000000 rcv_urg 00000000 rcv_nxt 00000000
     rcv_end 00000000
iss 1E3202D4 seq_fin 1E3202D4 snd_end 1E3202D4
      snd_urp 00000000 snd_lst 00000000

snd_nxt 1E3202D4 snd_una 1E3202D4 snd_wl 00000000
                                  snd_hi 1E3202D4

rex_val 00000000 rtl_val 00000000 xmt_val 00000000
flags:
snd_wnd 0  maxseg 0  xmtime 2  rxtct 0
timers:
INIT 0  REXMT 0  REXMTTL 0  PERSIST 0  FINACK 0
t_rcv_next 1A9A50  t_rcv_prev 1A9A50
```

NAME

edns – invoke editor for **ns_helper**

SYNOPSIS

/etc/edns [[*net.*]*node_id*]

DESCRIPTION

edns allows you to inspect and/or modify **ns_helper**'s master network root directory and replica list. Once invoked, **edns** enters an interactive mode and accepts the commands described in **help edns** commands.

[*net.*]*node_id* (optional)	Set the default **ns_helper** to the **ns_helper** at the node specified by the internet address.	
	Default if omitted:	Set the default **ns_helper** to any active **ns_helper**. An **ns_helper** becomes active after its database has been initialized.

SEE ALSO

More information is available. Type

help edns commands	For a summary of **edns** commands

NAME

/etc/**edrgy** – edit the network registry database

SYNOPSIS

/etc/**edrgy** [**–a** | **–p** | **–g** | **–o**] [**–l**] [**–s** *//site*] [**–synch**] [**–v**]

DESCRIPTION

The **edrgy** tool views and edits information in the registry database. You can invoke **edrgy** from any node.

Though anyone can read information in the registry database, you can usually change information only if you own the affected database entries. For example, only the owner of a group can add a name to the group's membership list.

With **edrgy**, you can edit and view names, accounts, and policies in the network registry, as well as entries in the local registry. The tool operates in one of four domains: person names, group names, organization names, and accounts.

OPTIONS

You can specify only one of **–a**, **–p**, **–g**, and **–o**.

–a (default)	Edit or view accounts.
–p	Edit or view persons.
–g	Edit or view groups.
–o	Edit or view organizations.
–l	Edit or view entries in local registry.
–s	Use the specified registry site.
–synch	Synchronize local registry with network registry.
–v	View selected entries.

Unless you specify the –v option, **edrgy** operates interactively. The following sections describes the commands you can enter in the interactive mode.

COMMANDS FOR PERSONS, GROUPS, AND ORGANIZATIONS

v[iew] [*name* | *number*] [**–f**] [**–m**] [**–po**]

View *name* entries.

If you specify a *number*, **edrgy** displays all matching entries, including any aliases.

The **–f** option displays entries in full (all fields except the membership list and organization policy).

If you are viewing groups or organizations, **–m** displays the membership list. For persons, **–m** lists all groups of which the person is a member, including groups that cannot appear in a project list.

If you specify **–po** while viewing organizations, **edrgy** displays policy information. Otherwise, it shows only the name and the UNIX number.

a[dd] [*person number* [*fullname*] [**–al**] [**–o** *owner*]]
a[dd] [*group number* [*fullname* [*password*]] [**–nl**] [**–o** *owner*]]
a[dd] [*organization number* [*fullname* [*password*]] [**–o** *owner*]]

Create a new name entry.

If you do not specify a *person*, *group*, or *organization* name, the **add** command enters an interactive mode and prompts you for each field in the entry. If you are adding organizations in the interactive mode, the command prompts you for policy information as well.

Specify the *owner* as a *person.group.organization* triplet. You can use *%* as a wildcard for any or all of the components. If you do not use the **–o** option, **edrgy** assigns the default owner, which you can set or display with the **defaults** command.

For persons, the **–al** option creates an alias entry. If *number* (the UNIX number) is already assigned to a person and you do not specify **–al**, an error occurs and you must either choose a different *number* or specify **–al**. If you use **–al** to create an alias and *number* is not already associated with a primary name, **edrgy** issues a warning but creates the alias.

For groups, the **–nl** flag indicates that the group is not to be included on project lists; omitting this flag allows the group to appear on project lists.

For groups and organizations, a space between quotation marks indicates a nil password.

Use quotation marks to embed spaces (or quotation marks) in a *fullname*. A single space between quotation marks indicates a nil *fullname*.

c[hange] [*person* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–al** | **–pr**]]
c[hange] [*group* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–p** *password*] [**–nl** | **–l**]]
c[hange] [*organization* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–p** *password*]]

Change a *name* entry.

If you do not specify a *person*, *group*, or *organization* name, the **change** command enters an interactive mode and prompts you for a name. If you do not specify any fields, the command prompts you for each field in succession. To leave a field unchanged, press <RETURN> at the prompt. If you are changing organization entries in the interactive mode, the command prompts you for policy information as well.

For person entries, the **–al** flag changes a primary name into an alias, while the **–pr** flag changes an alias into a primary name. This change can be made only from the command line, not in the interactive mode.

For group entries, the **–nl** flag disallows the group from appearing in project lists, while the **–l** flag allows the group to appear in project lists.

For organization entries, you can change policy information only in the interactive mode.

A single space between quotation marks indicates a nil *fullname* or *password*.

Specify the *owner* as a *person.group.organization* triplet. You can use *%* as a wildcard for any or all of the components.

Changes to a person name are reflected in membership lists that contain the person name. For example, if the person **ludwig** is a member of the group **composers** and the person name is changed to **louis**, the membership list for **composers** is automatically changed to include **louis** but not **ludwig**.

Changes to *number* (the UNIX number) cause the operating system to change its mapping of the UID, the primary name, and any aliases from the old *number* to the new one. However, files owned by the old *number* do not automatically show the new *number* as their owner.

The only fields of reserved entries that you can change are the *fullname*, the *password*, the *owner*, and (for *groups*) the property that allows a *group* to appear in project lists. If a reserved *group* is allowed to appear in project lists, you can disallow it; but if the *group* is disallowed, you cannot allow it.

m[ember] [*group* | *organization* [**–a** *member_list*] [**–r** *member_list*]]

Edit the membership list for a group or organization.

If you do not specify a group or organization, the **member** command enters an interactive mode and prompts you for names to add or remove.

The **–a** flag precedes the person names (separated by spaces) to be added to the membership list, while the **–r** flag precedes those to be removed. If you do not include either flag on the command line, **edrgy** prompts you for names to add or remove.

Adding a person to a membership list permits creation of a login account for that person with that group or organization.

Removing *person* from the membership list for *group* has the side effect of deleting all login accounts of the form *person.group*, and likewise for organizations.

del[ete] { *person* | *group* | *organization* }

Delete a name entry.

You cannot delete reserved names. Deleting a group or organization has the side effect of deleting any accounts with that group or organization.

adopt *uid_high.uid_low person number* [*fullname*] [**–o** *owner*]
adopt *uid_high.uid_low group number* [*password* [*fullname*]] [**–nl**] [**–o** *owner*]
adopt *uid_high.uid_low organization number* [*password* [*fullname*]] [**–o** *owner*]

Create a primary name entry for the specified UID.

The UID must be an orphan (a UID for which no name exists in any domain). The *uid_high* and *uid_low* are hexadecimal numbers.

An error occurs if you specify a name or UNIX number that is already defined within the same domain of the database.

A single space between quotation marks indicates a nil *fullname* or *password.*

Specify the *owner* as a *person.group.organization* triplet. You can use *%* as a wildcard for any or all of the components. If you do not use the **–o** option, **edrgy** assigns the default *owner*, which you can set or display with the **defaults** command.

COMMANDS FOR ACCOUNTS

In all of the account operations, the *account* argument is a *person.group.organization* triplet such as **jones.graphics.research**. Unless otherwise specified, any or all of the components can be the wildcard character, *%*. For example, **view** *%*.**dev**.*%* views all accounts associated with the group **dev**.

In an *account* argument, if you omit a trailing *organization* (or *group.organization*), *%* (or *%.%*) is assumed. Thus, **keats.%.%**, **keats.%**, and **keats** are equivalent.

v[**iew**] [*account*] [**–f**]

Display login accounts specified by the *account* pgo (*person, group, organization*) triplet.

Without the **–f** flag, **view** displays only the user fields in each account entry: abbreviated account S encrypted password, miscellaneous information, home directory, and login shell.

With **–f**, **view** displays the full entry, including the administrative fields as well as the user fields. Administrative information includes who created the account, when it was created, who last changed it, when it was last changed, when it expires, whether it is valid, whether the password is valid, and when the password was last changed.

a[**dd**] [*account* [**–a** { **p** | **pg** | **pgo** }] [*password* [*misc* [*homedir* [*shell*]]]]
[**–pnv**] [**–x** *account_exp* | **none**] [**–anv**]]

Create a login account.

Specify *account* as a pgo triplet. Wildcards are not allowed. If you do not supply an *account* on the command line, **add** enters an interactive mode and prompts you for each field in succession.

If the person specified in *account* is not already a member of the specified group and/or organization, **edrgy** automatically attempts to add the person to the membership lists. If you are not an owner of the group and/or organization, the attempt will fail and the account will not be created.

The **–a** flag indicates the degree of abbreviation allowed for login: **p** means that only the person is required; **pg** means the person and the group; **pgo** means that all three components of the account SID are required. (Of course, a user can always supply more components than are required.) If the abbreviation you specify is already defined for another account, **edrgy** automatically uses the shortest unique abbreviation and issues a warning.

For example, if you create an account **babar.elephants.none** with the abbreviation **p**, a user need only enter **babar** at the login prompt to use the account. If you then create an account **babar.kings.none**, the **p** abbreviation will conflict with the existing account, so the **pg** abbreviation, **babar.kings**, will be the shortest unique one.

Omitting the **–a** is equivalent to specifying **–a p** and results in use of the shortest unique abbreviation.

The *password* must adhere to the policy of the associated organization or the policy of the registry as a whole, whichever is more restrictive.

The *misc* field is not used by the operating system. The *gecos* field of each account's entry in the /etc/**passwd** file is the concatenation of the person's full name and the account's *misc*. Use quotes to include spaces, hyphens, or quotes in *misc*.

The *homedir* and *shell* are pathnames. The default *homedir* is /. The default *shell* is the null string.

Use a single space between quotation marks to indicate a nil *password*, *misc_info*, *homedir*, or *shell*.

The **–pnv** (password not valid) flag specifies that at the next login (for a newly created account, the first login), the user must change the password. If you omit this option, the password is valid.

The –x flag sets an expiration date for the account; the default is none.

The **–anv** (account not valid) flag specifies that the account is not currently valid for login. If you omit this option, the account is valid.

c[hange] [*account* [**–n** *new_account*] [**–a** { **p** | **pg** | **pgo** }]
[–p *password*] [**–m** *misc*] [**–h** *homedir*] [–s *shell*]
[**–pnv** | **–pv**] [–x *account_exp* | **none**] [**–anv** | **–av**]

Change one or more account entries.

Specify *account* as a pgo triplet. Wildcards are allowed, unless you use the **–n** option. If you do not supply an *account* on the command line, **change** enters an interactive mode and prompts you for each field in succession. Press <RETURN> to leave a field unchanged.

The command line arguments are largely the same as those of the **add** command. The **–n** flag enables you to change the account SID to *new_account*, a pgo triplet that cannot contain wildcards. The **–pv** flag specifies that the password is valid. The **–av** flag specifies that the account is valid.

You can enter a single space between quotation marks to indicate a nil *password*, *misc*, *homedir* or *shell*.

del[ete] *account*

Delete the entry for *account*, a pgo triplet that cannot contain wildcards.

MISCELLANEOUS COMMANDS

do[main] [p | g | o | a]

Change or display the type of registry information being viewed or edited.

You can specify **p** for persons, **g** for groups, **o** for organizations, or **a** for accounts. If you supply no argument, **edrgy** displays the current domain.

s[ite] [*//site*] [**–l**]

Change or display the registry site being viewed or edited.

If you specify a *//site*, **edrgy** attempts to use the registry server at the named site. If you specify **-l**, **edrgy** uses the local registry. If you supply no argument, **edrgy** displays the current site.

prop[erties]

Change and/or display the registry properties and policies.

This command prompts you for any changes to make. Press <RETURN> to leave information unchanged.

synch[ronize]

Update the local registry to match the master registry.

If a matching entry cannot be retrieved from the network registry, the local entry is marked invalid for login, and its UNIX numbers are updated.

co[py] [*account*]

Copy information for the specified accounts from the master registry to the local registry.

The *account* is a pgo triplet that can contain wildcards; trailing wildcard components can be omitted. If a matching account already exists in the local registry, **edrgy** updates the information to match that in the master registry; otherwise, **edrgy** adds the entry. If all entries in the local registry are used, **copy** reports an error and terminates.

def[aults]

Change and/or display the default values that **edrgy** uses.

h[elp] [*command*]
: Display usage information for **edrgy**.

  If you do not specify a particular command, **edrgy** lists the available commands.

q[uit]
: Exit **edrgy**.

COMMANDS VALID FOR THE LOCAL REGISTRY

To edit or view the local registry, use the **–l** flag when you invoke **edrgy**. This section lists the commands that are valid for editing or viewing the local registry. Unless otherwise specified, all options are as described in the previous command descriptions.

v[iew] [*name* | *number*] [**–f**] [**–po**]
: View name entries. (The **–m** option is not valid.)

v[iew] [*account*] [**–f**]
: Display specified login accounts.

c[hange] [*account* [**–a** { **p** | **pg** | **pgo** }] [**–m** *misc*] [**–h** *homedir*] [**–anv**]
: Change one or more account entries. (The **–p**, **–s**, **–pnv**, **–pv**, **–x**, and **–av** options are not valid.)

del[ete] *account*
: Delete an account entry.

do[main] [**p** | **g** | **o** | **a**]
: Change or display the type of registry information being viewed or edited.

s[ite] [*//site*] [**–l**]
: Change or display the registry site being viewed or edited.

prop[erties]
: Change and/or display the registry properties and policies.

synch[ronize]
: Update the local registry to match the master registry.

co[py] [*account*]
: Copy information for the specified accounts from the master registry to the local registry.

def[aults]
: Change and/or display the default values that **edrgy** uses.

h[elp] [*command*]
: Display usage information for **edrgy**.

q[uit]
: Exit **edrgy**.

NAME

find_orphans – locate and catalog uncataloged objects

SYNOPSIS

/etc/**find_orphans** [*options*] [*volume_pathname*]

DESCRIPTION

find_orphans finds all uncataloged permanent objects in a local volume. It uses or creates a directory "lost+found" in the root of the volume and creates entries for all objects not cataloged elsewhere.

find_orphans can operate by itself by using search mode, in which case it searches the volume for all orphans, or it works with **salvol** (list mode, the default), in which case it just catalogs the orphans detected by the previous run of **salvol**. Both methods find exactly the same set of orphans. We recommend that you run **find_orphans** every time a node is booted, and on every mounted volume. Below is a description of the two modes of operation.

The objects cataloged by **find_orphans** are given sequential names like f1, f2, etc., and you can move them using **/com/mvf** or **/bin/mv** to a directory of your choice. **find_orphans** is useful for finding objects that were being updated during a system crash or that were uncataloged through program errors.

In list mode (the default), **find_orphans** catalogs all objects listed in the file **lost+found.list** in the root directory of the volume. If the file does not exist, **find_orphans** creates it. Note that **invol** creates the file when it creates the volume. If the **lost+found.list** exists, **salvol** enters information describing each orphan. List mode is considerably faster than search mode since there is no need to search the entire volume. You must have permission to catalog objects in **lost+found**.

In search mode, you must have read permission to all directories on the volume. If some directory is not readable, every object under that directory is cataloged in the **lost+found** directory. In addition, you must have permission either to create the **lost+found** directory or to catalog objects in **lost+found** when it already exists.

Search mode should be run only on a quiescent node; that is, one not connected to the network (use **netsvc –n** to disable network communications) and not actively running any processes other than the one performing the **find_orphans** operation.

volume_pathname (optional)

Specify the name of the volume to be searched. The volume must be physically attached to your node; you may not find orphan objects on volumes elsewhere in the network.

Default if omitted: search node boot volume

OPTIONS

–l[ist] (default)	List mode, catalog objects listed in **/lost+found.list**.
–s[earch]	Search mode, search the volume for orphans.
–v[erify]	Verify only; don't catalog any orphans.

EXAMPLES

```
$ /etc/find_orphans
Cataloging in: /lost+found
39D40FF0.100086CA  ->  f1
39D40F9D.E00086CA  ->  f2
39D41026.600086CA  ->  f3
39D40DA6.D00086CA  ->  f4
39D40998.200086CA  ->  f5
39D41042.800086CA  ->  f6
39D40CB8.E00086CA  ->  f7
39D41001.300086CA  ->  f8
39D40F7E.D00086CA  ->  f9
39D40CCE.F00086CA  ->  f10
39D40D8B.C00086CA  ->  f11
39D40E33.100086CA  ->  f12
39D40A06.700086CA  ->  f13
39D40F23.900086CA  ->  f14
39D40E16.000086CA  ->  f15
39D40F36.A00086CA  ->  f16
39D41C0A.200086CA  ->  f17
Number of orphans catalogued: 17
```

```
$ ld -a /lost+found

Directory "/lost+found":

sys   type      blocks  current
type  uid         used   length  attr rights       name

file  uasc           1       32  P    prwx-        f1
file  unstruct       0        0  P    prwx-        f10
file  uasc           1       32  P    prwx-        f11
file  unstruct       0        0  P    prwx-        f12
file  unstruct       1       54  P    prwx-        f13
file  uasc           1       32  P    prwx-        f14
file  uasc           1       32  P    prwx-        f15
file  unstruct       0        0  P    prwx-        f16
file  unstruct       0        0  P    prwx-        f17
file  unstruct       0        0  P    prwx-        f2
file  uasc           1       32  P    prwx-        f3
file  unstruct       0        0  P    prwx-        f4
file  coff           1   101376  P    prwx-        f5
file  unstruct       0        0  P    prwx-        f6
file  uasc           1       32  P    prwx-        f7
file  unstruct       0        0  P    prwx-        f8
file  uasc           1       32  P    prwx-        f9

17 entries, 9 blocks used.
```

NAME

ftpd – DARPA Internet File Transfer Protocol server

SYNOPSIS

/etc/**ftpd** [**–d**] [**–l**] [**–t***timeout*]

DESCRIPTION

ftpd is the DARPA Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the "ftp" service specification; see **services.**

OPTIONS

–d	Write debugging information to the syslog.
–l	Log each **ftp** session in the syslog.
–t*timeout*	Set the inactivity timeout period to *timeout*. Without this option, the **ftp** server will timeout an inactive session after 15 minutes.

FTP REQUESTS

The **ftp** server currently supports the following **ftp** requests; case is not distinguished.

Request	**Description**
ABOR	Abort previous command
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	List files in a directory ("ls -lg")
MKD	Make a directory
MODE	Specify data transfer *mode*
NLST	Give name list of files in directory ("ls")
NOOP	Do nothing
PASS	Specify password
PASV	Prepare for server-to-server transfer
PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from filename
RNTO	Specify rename-to filename
STOR	Store a file
STOU	Store a file with a unique name
STRU	Specify data transfer *structure*

TYPE	Specify data transfer *type*
USER	Specify username
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The remaining **ftp** requests specified in Internet RFC 959 are recognized, but not implemented.

The **ftp** server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959.

ftpd interprets filenames according to the ''globbing'' conventions used by **csh**(1). This allows users to utilize the metacharacters ''*?[]{}~''.

ftpd authenticates users according to four rules.

1) The username must be in the password data base, **/etc/passwd**, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.

2) The username must not appear in the file /etc/**ftpusers**.

3) The user must have a standard shell.

4) If the username is ''anonymous'' or ''ftp'', an anonymous ftp account must be present in the password file (user ''ftp''). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, **ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot** command to the home directory of the ''ftp'' user. In order that system security is not breached, we recommend that the ''ftp'' subtree be constructed with care; the following rules are recommended.

~ftp	Make the home directory owned by ''ftp'' and unwritable by anyone.
~ftp/bin	Make this directory owned by the super-user and unwritable by anyone. The program **ls**(1) must be present to support the list commands. This program should have mode 111.
~ftp/etc	Make this directory owned by the super-user and unwritable by anyone. The files **passwd** and **group** must be present for the **ls** command to work properly. These files should be mode 444.
~ftp/pub	Make this directory mode 777 and owned by ''ftp''. Users should then place files which are to be accessible via the anonymous account in this directory.

BUGS

The anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

NAME

gettable – get NIC format host tables from a host

SYNOPSIS

/etc/gettable [–v] *host* [*outfile*]

DESCRIPTION

gettable is a simple program used to obtain the NIC standard host tables from a ''nicname'' server. The indicated *host* is queried for the tables. The tables, if retrieved, are placed in the file *outfile* or by default, **hosts.txt**.

gettable operates by opening a TCP connection to the port indicated in the service specification for ''nicname''. A request is then made for ''ALL'' names and the resultant information is placed in the output file.

gettable is best used in conjunction with the **htable** program which converts the NIC standard file format to that used by the network library lookup routines.

OPTIONS

–v	Get just the version number instead of the complete host table, and puts the output in the file *outfile* or by default, **hosts.ver**.
outfile	Place the tables in the specified *outfile*.

BUGS

If the name-domain system provided network name mapping well as host name mapping, **gettable** would no longer be needed.

SEE ALSO

htable, named;
Configuring and Managing TCP/IP.

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/etc/ifconfig *interface [address_family*] [*address* [*dest_address*]] [*parameters*]
/etc/ifconfig *interface* [*protocol_family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine. It may also be used at a later time to redefine an interface's address or other operating parameters. The *interface* parameter is a string of the form *name unit*, for example, **eth0**.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, it is necessary to specify the *address_family*, which may change the interpretation of the remaining parameters. The only address family currently supported by Apollo is **inet .**

For the DARPA-Internet family, the address is either a host name present in the host name data base, **hosts**, or a DARPA Internet address expressed in the Internet standard ''dot notation''.

PARAMETERS

The following parameters may be set with **ifconfig**:

up Mark an interface ''up''. This may be used to enable an interface after an ''ifconfig down.'' It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down Mark an interface ''down''. When an interface is marked ''down'', the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers Request the use of a ''trailer'' link level encapsulation when sending (default). If a network interface supports **trailers**, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see **arp**; currently, only 10 MB ETHERNET*), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only.

–trailers	Disable the use of a ''trailer'' link level encapsulation.
arp	Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10MB ETHERNET addresses.
–arp	Disable the use of the Address Resolution Protocol.
metric *n*	Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (**routed**). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.
debug	Enable driver dependent debugging code; usually, this turns on extra console error logging.
–debug	Disable driver dependent debugging code.
netmask *mask*	(Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table **networks**. The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.
dstaddr	Specify the address of the correspondent on the other end of a point to point link.
broadcast	(Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

NOTES

ETHERNET is a registered trademark of the Xerox Corporation.

SEE ALSO

netstat, rc;
Configuring and Managing TCP/IP.

NAME

lcnet – display internet routing information

SYNOPSIS

/etc/lcnet [*options*]

DESCRIPTION

lcnet displays the list of known networks, their distance from the current node, the router used as the first hop from that network, and other information.

The distance (hops) from remote networks is measured in intervening routers. The distances are all for one-way traffic; if a network is three hops away from yours, your requests pass through three routers to get to that network. The responses also pass through three routers on the way back.

The **–conn** option shows you the full internet topology; that is, the list of networks and how the routers connect them together.

OPTIONS

–local (default)
: Display the "First Hop" and "Hops" information for each network in the internet. The first hop is the node ID of a router on your network. It is the first router used in sending packets from your network to the target network. Other routers are also used if the target network is more than one hop away from your own.

–full
: Display information showing how up to date the routing table is (the "Age" and "Expiration" columns) in addition to the "First hop" and "Hops" information shown by the **–local** option. **–full** also lists inaccessible networks.

–conn
: Show which routers are connected to each network, and which other networks those routers touch. This option displays the "Touching" information.

–hw
: Display the type of hardware used for each of the networks (ring or IIC).

: The **–conn** and **–hw** options may take several seconds to execute if you have a large internet.

–n *node-spec*
: Print another node's view of the internet. The outputs produced by **–local** and **–full** vary from node to node; **–n** affects these outputs. The **–n** option does not affect the output produced by the **–conn** or **–hw** options, since the hardware and connectivity do not depend on a node's position in the internet.

–c The **–c** option suppresses the title over each output column. It also fills every line of the "Network" column produced by the **–conn** option, and every line of the "Hardware" column produced by the **–hw** option. These format changes make it easier to use **lcnet**'s output as another program's input.

EXAMPLES

In this example, the node is directly connected to network C0FFEE. Networks 5A1AD and ED1F1CE were connected in the past, but are not now accessible (perhaps because the routers are down).

The expiration date and time for the "local" network are meaningless.

```
$ /etc/lcnet –full
            First
Network      Hop     Hops    Age  Expiration date/time
========    =====   =====    ===  ===================
    B020     4B6F       1    NEW  1987/06/16 14:33:21
  B00B00     4B6F       2    NEW  1987/06/16 14:33:21
   5A1AD     4B6F    gone    NEW  1987/06/16 14:33:21
  C0FFEE        0   local    NEW  1987/06/09 10:27:46
 ED1F1CE     4B6F    gone    NEW  1987/06/16 14:33:21
    D0D0     BAD1       1    NEW  1987/06/16 14:33:39
```

The "Touching" information describes your internet completely. This example includes several kinds of information:

```
- Network DEFACED has one router, node 2A3B.
  That router connects DEFACED to EFFACED.
- Network FACE0FF contains two routers, 31DC and 1371.
  Those routers connect FACE0FF to C0C0A and C0FFEE,
  respectively.
```

$ /etc/lcnet –conn

```
              Touching      Touching
Network        Router       Network
========     ========     ========
    F00D         5C0B        DECAF
                 36CF       C0FFEE
   5A1AD         459B       C0FFEE
                 45BE      ED1F1CE
   B002E         3F0A       C0FFEE
   C0C0A         BAD1       B00B1E
                 56B0      EFFACED
                 31DC      FACE0FF
   DECAF         5C0B         F00D
  B00B1E         BAD1        C0C0A
  C0FFEE         36CF         F00D
                 459B        5A1AD
                 3F0A        B002E
                 1371      FACE0FF
 DEFACED         2A3B      EFFACED
 ED1F1CE         45BE        5A1AD
 EFFACED         56B0        C0C0A
                 2A3B      DEFACED
 FACE0FF         31DC        C0C0A
                 1371       C0FFEE
```

$ /etc/lcnet –conn –c

```
    F00D         5C0B        DECAF
    F00D         36CF       C0FFEE
   5A1AD         459B       C0FFEE
   5A1AD         45BE      ED1F1CE
   B002E         3F0A       C0FFEE
   C0C0A         BAD1       B00B1E
   C0C0A         56B0      EFFACED
   C0C0A         31DC      FACE0FF
   DECAF         5C0B         F00D
  B00B1E         BAD1        C0C0A
  C0FFEE         36CF         F00D
  C0FFEE         459B        5A1AD
  C0FFEE         3F0A        B002E
  C0FFEE         1371      FACE0FF
 DEFACED         2A3B      EFFACED
 ED1F1CE         45BE        5A1AD
 EFFACED         56B0        C0C0A
```

```
    EFFACED         2A3B      DEFACED
    FACEOFF         31DC        COCOA
    FACEOFF         1371       COFFEE
```

SEE ALSO

More information is available. Type

help lcnode	For information on listing connected nodes

NAME

lcnode – list nodes connected to the network

SYNOPSIS

/etc/lcnode [*options*]

DESCRIPTION

lcnode lists the nodes currently connected to the network. The list contains the ID of every node connected, the time at which the node was started, the current time, and the name of each node's entry directory.

This command reports only the nodes that respond within a preset time limit. If a node is connected, but temporarily unable to respond within the specified time, it does not appear in the produced list.

OPTIONS

–m[e]	Request information about your node only. This option displays the node ID.
–b[rief]	Request brief output. **lcnode** lists only the entry directory name for each connected node. Note that the entry directory of a diskless node is the entry directory of its paging partner.
–id	When used with **–brief**, display the node ID in addition to the entry directory.
–c[ount]	Request node count only. **lcnode** lists only the number of nodes responding to its query.
–max[nodes] *n*	Set a limit on the number of nodes you want to see, even if more could respond.
–from *node_spec*	Starts the node list at some node other than your own. This is especially useful in an internet environment, for looking at networks other than your own. See **help node_spec** for details about node specification syntax.
–name	When you specify the **–brief** option, **lcnode** normally prints the entry directory for each node. If you specify **–name** with **–brief**, **lcnode** prints the node name cataloged with the naming server. Only diskless nodes are printed differently. A diskless node's entry directory is its partner's node name; a diskless node's node name is uniquely its own. Unless the **–from** option specifies your own node, the list includes only an unbroken sequence of nodes running Aegis SR9.0 or later. The rest of the node list is lost, starting with the first node running a pre-SR9.0 Aegis.

EXAMPLES

1. $ **/etc/lcnode**

```
The node ID of this node is 21.  3 other nodes responded.

id        Boot time         Current time      Entry Directory

21  1987/06/09  9:21:44 1987/06/09 16:06:22  //dollar
17  1987/06/09 13:52:02 1987/06/09 16:06:13  //quarter
27  1987/06/09 12:53:28 1987/06/09 16:06:07  //nickel
11  1987/06/09 12:03:39 1987/06/09 16:06:15  ** DISKLESS **
                                    //diskless_$11 partner node: 17
```

2. $ **/etc/lcnode –me**

```
     The node id of this node is 21.
```

3. $ **/etc/lcnode –b**

```
    //dollar
    //quarter
    //nickel
    //quarter

  (//quarter appears once as the host for a diskless node and
   once for the node with the disk.)
```

4. $ **/etc/lcnode –b –name**

```
    //dollar
    //quarter
    //nickel
    //diskless_$000011

   (–name shows you the name under which diskless node 11
   is cataloged)
```

5. $ **/etc/lcnode –c**

```
    466 other nodes responded.
```

6. $ **/etc/lcnode –c –b**

```
    466
```

7. $ **/etc/lcnode –c –m**
```
The node id of this node is 116A.
466 other nodes responded.
```

8. $ **/etc/lcnode –b –id**
```
21  //dollar
17  //quarter
27  //nickel
11  //quarter
```

9. $ **/etc/lcnode –from 0FAD.3924 –max 2**
```
 Starting from node 3924.
 1 other node responded,
    but more might have responded with a high -max value.

Node id  Boot time              Current time         Entry Directory

 3924   1985/02/14 17:20:45   1985/02/14 19:07:04  //laurel
 34Bf   1985/02/14 18:46:52   1985/02/14 19:08:09  //hardy
```

SEE ALSO

More information is available. Type

help lcnet	For information on listing connected networks in an internet environment

NAME

mbd – dump usage info on tcp buffer pool

SYNOPSIS

/etc/mbd [**–f**] [**–k**]

DESCRIPTION

The **mbd** command dumps usage information about the tcp memory buffer pools. Usage statistics on tcp memory buffers may be obtained by using the **-m** option of the **netstat** command; **mbd** is intended for analyzing cases where buffers are being lost. It scans the entire buffer pool, finding all the chains of in-use buffers; it then prints each chain of buffers. This information may help you in discovering reasons why buffers are being lost.

OPTIONS

–f Dump the free pools as well as the chains of in-use buffers. This produces a lot of output.

–k Don't try to lock the mutex on the buffer pools before doing the dump. This is useful mostly when the **tcpd** has crashed with the buffer pool mutex locked.

EXAMPLES

A dump of the buffer pools of a basically idle tcp might look like this:

```
$ /etc/mbd
Offset 0x000035cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x000041cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00003bcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x0000a7cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00004dcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00007dcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
```

Here there are 6 large (1520-byte) buffers in use, all on a single chain.

NAME

netmain – analyze network maintenance stats

SYNOPSIS

/etc/netmain [*options*]

DESCRIPTION

netmain is an interactive, menu-driven program that lets you control the **netmain_srvr**, the network maintenance server, and analyze the data that **netmain_srvr** produces. **netmain** provides detailed help from its menus.

OPTIONS

–w[**help**] (default)	Make sure the window is large enough to display command menus and interactive help.
–wc[**md**]	Set the window size smaller for command menus only. If you later decide that you want to see the helps, grow the window manually with <GROW>.
–nw	Do not change the window size.

EXAMPLES

1. Run **netmain** in a window large enough to display command menus and interactive help:

```
$ /etc/netmain
```

2. Run **netmain** in a window large enough (but no larger) to display the command menus:

```
$ /etc/netmain –wc
```

SEE ALSO

More information is available. Type

help netmain_srvr	For details about gathering network error statistics
help netmain_note	For information about adding notes to the network error log

NAME

netmain_chklog – clean up bad log files

SYNOPSIS

/etc/**netmain_chklog** *[options]* pathname...

DESCRIPTION

When the **netmain_srvr** program halts catastrophically (for instance, during a node reset), it can leave the log file it was writing in a corrupt, unusable state. **netmain_chklog** determines whether the log is corrupt and, optionally, deletes corrupt files.

If the pathname you specify points to some kind of file other than a **netmain** log file, that file is almost always ignored; it is almost never deleted as a corrupt log. On very rare occasions, another kind of file may look so much like a corrupt log that it might be deleted accidentally if you use both **–d** and the standard command option **–nq** (no query). Thus you should use **–d –nq** with extreme care.

pathname (required)	Specify the files to be checked. Multiple names and wildcarding are permitted; separate names with blanks.

OPTIONS

–d	Delete corrupt log files.
–nd (default)	Do not delete anything.
–l (default)	Describe every file analyzed.
–nl	Describe only corrupt log files.

EXAMPLES

```
$ /etc/netmain_chklog 'node_data/net_log/?*
```

SEE ALSO

More information is available. Type

help netmain_srvr	For details about **netmain**'s data collection server

NAME

netmain_note – place message in network error log

SYNOPSIS

/etc/netmain_note *string* [*string* ...]

DESCRIPTION

netmain_note sends a text string to **netmain_srvr**, the network maintenance server. The message is broadcast to all maintenance servers.

Typical topics of maintenance notes include known or explainable network failures, scheduled down-time, and node installations.

string (required)	Specify message to be sent. You may use any string that is legal in a shell command. (Note that the shell takes special action on some keywords, such as 'if', unless you place them in quotation marks.) If there is more than one string, **netmain** builds the note by concatenating the arguments that are separated by spaces.

EXAMPLES

```
$ /etc/netmain_note 'Scheduled down time at 5 pm.'
```

```
$ /etc/netmain_note Cable disconnected at //sancho_panza
```

SEE ALSO

More information is available. Type

help netmain	For instructions for controlling **netmain_srvr** after it starts, and for analyzing the data it collects
help netmain_srvr	For details about **netmain**'s data collection server

NAME

netmain_srvr – collect network error stats

SYNOPSIS

/etc/netmain_srvr [options] [*pathname*]

DESCRIPTION

netmain_srvr collects and stores performance statistics for the Apollo token ring network. Use **netmain_srvr** to gather information; use the **netmain** program to display and analyze the information.

You can set parameters for **netmain_srvr** from the command line and from an options file. Once the server is running, you can change any parameter using the **netmain** program. To include parameters in an options file, specify the **–cmdf** option.

When you specify **–cmdf** *pathname*, **netmain_srvr** reads the options listed in the options file first and then reads any other options on the **netmain_srvr** command line. If options specified in the file and on the command line differ, **netmain_srvr** uses the command line settings. For example, if the options file specifies a log file length as **–ll 1500**, and the command line specifies **–ll 3000**, **netmain_srvr** uses **–ll 3000**.

If a **netmain_srvr** does not start properly, a record of the failure appears in **'node_data/netmain_srvr.err_log**.

OPTIONS

–a[ppend]	Append to an existing log file with this name, if one already exists; otherwise, create a log file with this name. This option is only valid when a log file pathname is specified with the **–l** option. Contrast this with the **–nappend** option.
–cmdf *pathname*	Accept options from an ASCII text file *pathname*. You may use this option only from the command line, not in the options file. There can only be one options file.
–l[og] *[pathname]* (default)	Create a log file. Optionally, specify a pathname, which is relative to the **'node_data/net_log** directory. If either this option or the pathname is not specified, the log file name is derived from the current date: **'node_data/net_log/net_log.yy.mm.dd.** The log file is stored on the disk of the node running **netmain_srvr**, and must remain there for **netmain_srvr** to write to it.
–ll *n* (default)	Set an upper limit on the length of the log file. The file size limit *n* is in 1024-byte units. The default value for *n* is 3000. You must use this option when you start the monitor and if you don't want to use the default length for the first log file, since you cannot change the name of a log file once it's open.

–na[ppend] (default)
: Create a new log file, over-writing any log with that name, if one exists. This option is only valid when a log file pathname is specified with the **–l** option. Contrast this with the **–append** option.

–nl[log]
: Do not write to a log file. **netmain_srvr** still runs probes and observers.

–ntopo[_init] (default)
: Override the **–topo_init** option, if **–topo_init** is specified in an options file. **–ntopo** is useful only on the command line.

–obs[erve] *observer time ...*
: Set the interval at which the named observer wakes up. Specify *time* as hh:mm:ss, hh:mm, or *never*, if you do not want the monitor to run the observer. Multiple observer/time pairs are permitted. See the default times listed below for each observer.

–re_obs[erve] *observer time ...*
: Set the "Recheck interval" — the interval that the observer waits before rechecking a node that has caused an alarm condition. By setting a recheck interval, you ensure that the observer only reports on a node once every *time* period. If the recheck interval is too short, the observer may produce many redundant alarms. Specify *time* as hh:mm:ss, or hh:mm. Multiple observer/time pairs are permitted. See the default recheck intervals listed below for each observer.

–s[ample] *probe time ...*
: Set the interval at which the named probe wakes up. Specify *time* as hh:mm:ss, hh:mm, or **never**, if you do not want the monitor to run the probe. Multiple probe/time pairs are permitted. See the default times listed below for each probe.

–sk[ip] *probe distance ...*
: Set the skip distance for the probe named. If the skip distance is *n*, the named probe samples approximately 1/*n* of the nodes every time it wakes up. Multiple probe/distance pairs are permitted. See the default skip distances listed below for each probe.

–topo[_init] *pathname*
: Initialize the monitor's total node list from a data file. The file may contain any number of node names or hexadecimal IDs, separated by spaces or on separate lines. If there is a "#" or "{" in any line, that character and all characters to the right of it are ignored (that is, "#" and "{" are comment markers).

EXAMPLES

1. Command: **cps /etc/netmain_srvr –ll 1500 –l tuesday_error_log**

2. Command: **cps /etc/netmain_srvr –s err_counts 0:15 hw_fail never**

3. Command: **cps /etc/netmain_srvr –cmdf /etc/start.net_srvr –ll 3000**

```
{ The file /etc/start.net_srvr might contain        }
{ these lines:                                      }
{      –l –ll 1500                                  }
{      –sample err_counts 0:01:00 –skip err_counts 30  }
{      –sample topology 0:20:00                     }
{      –sample disk_errs 0:01:00 –skip disk_errs 30   }
{      –sample time_skew never                      }
{      –observe modem_errs 0:10:00                  }
```

NAME

netsvc – set or display network services

SYNOPSIS

/etc/netsvc [*options*]

DESCRIPTION

netsvc sets or displays the network services that this node will perform. All changes take place immediately.

OPTIONS

If no options are specified, **netsvc** displays the network services allowed for this node.

–n	None. Disable all network services and physically disconnect this node from the network.
–l	Local. Allow only network requests originating at this node.
–r	Remote. Allow only network requests originating at other nodes.
–a (default)	All. Allow both locally and remotely initiated network requests. (The size of the remote paging pool is not changed.)
–s[**ervers**] [*n*]	Servers. Set the number of network servers to run on this node. At system startup, the number of network servers is 1. If this node is a network partner for diskless nodes or has several remote file users, their performance can be improved by increasing the number of servers. If *n* is not specified, the maximum number of servers (3) is used.
–p [*n*]	Pool. Set local memory pool size. Network page requests originating at remote nodes may not use more than *n* pages of the local node's memory. If *n* is not specified, all the local node's memory is eligible for remote page requests.
–net [*net_id*]	Network ID. Set or display network ID. Use this option to change or examine the ID of the network to which the node is attached. It affects only the node at which you type the command, not the rest of the network. Specifying a hexadecimal network ID changes your node's network ID. Using **–net** with no argument forces **netsvc** to display your network ID even if it is set to 0. This option is useful only when there are no internet routers active on the node's network. Routers give the network ID to nonrouting nodes every 30 seconds, and may override the network ID you specify with this option.

NOTE

If the network ID you set with **–net** differs from the network ID used by other nodes on your network, your node may not be able to communicate with those other nodes.

Be careful when revoking network access with **–n** or **–l**. Remote file users may have problems, and writable files may be damaged. If your node was the network partner for a diskless node, that node will crash when your node leaves the network.

Use the –s option carefully. Although you can increase the number of servers, you cannot decrease it. The only way to return to a smaller number of servers is to reboot the node. Also note that increasing the number of server processes decreases the number of user processes allowed.

EXAMPLES

```
$ /etc/netsvc
Network operations allowed: ALL
Number of network servers: 1
Remotely initiated paging pool limit: NONE
Network ID: 437A9
$
```

SEE ALSO

More information is available. Type

help rtsvc	For information about controlling a node's internet routing service

NAME

obty – set or display the type of an object

SYNOPSIS

/etc/**obty** ([*object_type*] *pathname...*)

DESCRIPTION

obty is intended for system-level debugging use only. Misuse of this command can cause objects to become inaccessible and programs to behave incorrectly.

pathname (required)	Specify object whose type is to be set or displayed. Wildcarding of this pathname is permitted.
object_type (optional)	Specify new type setting. *object_type* must be a known type; the **lty** command lists the types currently defined on a volume. Executable files (output of compilers and binders) are **obj**, **coff** or **unstruct**. Most other binary files are **rec**. Default if omitted: display current type of *pathname*

EXAMPLES

The sequence of the following commands is significant.

Display current object type:

```
$ /etc/obty testfile
"testfile" object type is nil.
```

Set type to **uasc**:

```
$ /etc/obty testfile uasc
```

Display new object type:

```
$ /etc/obty testfile
"testfile" object type is uasc.
```

NAME

prf – queue a file for printing

SYNOPSIS

prf [*options*] *pathname...*

DESCRIPTION

The **prf** command queues a file for printing. The file must be an ASCII stream (that is, text) file, a graphics map file (GMF), or a GPR bitmap object. After successfully queuing a file, **prf** displays a message containing the full pathname of the file that you queued.

You can execute **prf** once for each file that you want to print (specifying all the necessary options every time), or you can enter **prf**'s interactive mode and hand files to the program continuously. See the examples for a sample interactive session.

Files queued by **prf** are physically printed by **prsvr**, the print server, running as a background task under control of **prmgr**, the print manager.

When you invoke **prf**, it first sets all options to their default states. Next, it looks for the print options file called **user_data/startup.prf** unless you invoke **prf** with the **–ndb** option. If **prf** locates the option file, it executes the options contained in the file to configure the current session. Finally, it processes all options on the command line.

pathname (optional)	Specify the file to be printed. Multiple pathnames and pathname wildcarding are permitted. Default if omitted: read standard input

OPTIONS

The following options can appear on the shell command line or in **prf** interactive mode. In addition, you can place one or more options in a **prf** option file so that they are executed automatically whenever you invoke **prf**.

Many of the options have default values that are specified in the **prsvr** configuration file established for each printer in the network by the system administrator. If you omit these options, your file is printed using the values specified in the **prsvr** configuration file. For example, omission of the **–banner** option could cause your file to be printed with a banner page if the **prsvr** configuration file specifies one.

If no options are specified, the file is printed using ASCII carriage control, with pagination enabled, on the default printer as established by the system administrator.

Options Applying to All File Types

–inter[active]
: Enter interactive mode.

–sea[rch_dir] {on|off}
: Search through all the directories of all the active processes on your node for the file(s) to be printed. This option is most useful in interactive mode, when the working directory of the **prf** process may be different from the working directory of the file to be printed.

  The default is **off**.

–cop[ies] *n*
: Print multiple copies of the file, where *n* is the requested number of copies. If **–cop[ies]** is specified, *n* is required. The default is one copy.

–pr[inter]*name*
: Specify the name of the printer that should print the file. This option is useful only if more than one printer is in use on the network, or if a printer has been assigned a nonstandard name with the **printer_name** configuration directive in the **prsvr** command. If you omit this option, **prf** uses the default printer name, **p**. Note that **p** is also the default printer name used by the print server.

–s[ite] *spool_node_name*
: Use this option only if you are queuing jobs to a pre-SR10 print server connected to a spool directory (/sys/**print**) that is different from the one specified by your node. By default, SR10 printers find the spool node for you.

–nc[opy]
: Print the specified file from its location in the user-specified directory, bypassing /sys/**print/spooler**. If you select this option, **prf** defaults to the no-delete (**–nd**) option. If you specify the delete (**–d**) option, the file is deleted at the completion of the print request. If you use this option (with or without the delete option), do not open and alter the print file before the print job is completed.

–d[elete] (default)
: Delete the print file at the completion of the print job.

–nd[elete]
: Do not delete the print file when the print server is finished printing it. This becomes default if **–nc** is specified.

–user[*username*]
: Specify the user name that appears on the banner page of the printed file. The alarm facility of **prf** also uses this name to determine who should be notified when printing is complete (see **–sig** below). This means that this name must be a valid log-in name (unless you don't care about sending an alarm).

  The default is the current log-in name.

–sig[nal] {alarm|off} Request an alarm server signal when the file has finished printing.

The default is **off**.

–ban[ner] [on|off] Enable/disable banner page. If the banner setting in the **prsvr** configuration file is **off**, no banner is printed.

The default is **on**.

–config[_file] [*pathname*]

Specify a file containing further **prf** options, one per line. Do not use prefixed hyphens (-) with the option names in the configuration file. If *pathname* is omitted, **prf** executes the **prf** option file **~/user_data/startup_prf**.

–ndb Suppress processing of the **prf** option file.

–trans[parent] [on|off]

off specifies that the file being printed is passed directly to the printer driver routine with no processing by the print server. The default is **on**.

–filter[_chain] *string*

Specifies a filter string that will be used by the print server to process the job. This option overrides the default processing done by the print server. It is most often used to invoke filters that have been added to the print server. The format of the string is "filter1 | filter2", where filter1 and filter2 are composed of strings of the form "type1$type2" and "type2$type3". Note that the output type of filter *n* must equal the input type of filter *n*+1 .

–paper_size {a|b|legal|a3|a4|a5|b4|b5}

Select the paper size. You must specify one of the following size codes:

Code	**Size in inches (mm)**
a	8.50 x 11.00
b	11.00 x 17.00
legal	8.50 x 14.00
a3	11.69 x 16.54 (297mm x 420mm)
a4	8.27 x 11.69 (210mm x 297mm)
a5	5.38 x 8.27 (137mm x 210mm)
b4	9.84 x 13.90 (257mm x 364mm)
b5	5.93 x 9.89 (182mm x 257mm)

This option is available only for the Domain/Laser-26 and APPLE LaserWriter* printers. Because **prf** assumes that the correct paper is in the printer's paper tray, you should check the paper tray before printing. The default paper size is specified in the **prsvr** configuration file.

Options Applying to Text Files Only

–margins [on|off] Enable/disable margins generated by **prf**.

The default is **on**.

–top *n* Top page margin, in inches. The default is a value specified in the **prsvr** configuration file.

–bot[tom] *n* Bottom page margin, in inches. The default is a value specified in the **prsvr** configuration file.

–right *n* Right margin, in inches. The default is 0 inches.

–left *n* Left margin, in inches. The default is 0 inches.

–headers [on|off] Enable/disable page headers and footers generated by **prf**. The default is **on**.

–head[_string] *l-string/c-string/r-string*

Specify contents of left, center, and right components of the page header generated by **prf**. Components can be empty strings. The following special characters return the values indicated when they appear in the header strings:

Character	Return Value
@	= Escape character
#	= current Page number with 1 leading and 1 trailing space
%	= Current date
!	= Filename
&	= Filename's last time, date modified
*	= Insert a space in text string (literal spaces are not allowed)

Example: **–head !/Page#/%** produces a header with the filename in the left component, the string "Page" followed by the current page number in the center component, and the current date in the right component. The default header is a string specified in the **prsvr** configuration file.

–foot[_string] *l-string/c-string/r-string*
Specify contents of page footers. The format is the same as for **–head** above. There is no default footer.

–ftn [on|off]
Enable/disable FORTRAN carriage control. **–ftn on** causes the print server to use FORTRAN forms control even if the file does not have the FORTRAN carriage-control flag. Use of this option causes **prf** to interpret the first character of each line as a FORTRAN carriage control character (and not print it). This can be unfortunate if the file has ASCII carriage control, so be careful. **–ftn off** causes the print server to print the contents of column one rather than trying to interpret it as FORTRAN forms control. If this option is specified without **on** or **off**, **on** is assumed.

The default is **off**.

–wrap [on|off]
Enable/disable automatic line wrapping. When enabled, **prf** wraps lines that exceed the right margin. When disabled, **prf** truncates lines that exceed the right margin. If this option is specified without **on** or **off**, **on** is assumed.

The default is **off**.

–col[umns] {1|2}
Specify single-or double-column printing.

The default state is single column.

–lpi *n*
Specify the line-spacing factor. *n* is an integer indicating the number of lines per inch.

The default is six lines per inch.

Options for Variable Font and Pitch

–pitch *n*
Set the printer pitch (characters/inch). The following pitch settings are available on the printers indicated.

Printer	**Pitch**
Printronix*	10
Spinwriter*	12
IMAGEN*	8.5, 10, 12, 15, 17.1
GENICOM*	10, 12, 13.1, 16.7
Versatec*	12
LaserWriter*	1 to 100
Laser-26	1 to 100

–point *n*	Set the point size for the font to be used. This is a real number that specifies size in points. A point equals 1/72 inch.
–weight {light|medium|bold}	Set the weight of the font to be used. The default is **medium.**
–lq [**on**|**off**]	Specify that the document is to be printed in letter quality (**on**) or in draft (**off**) mode. With no argument, **on** is assumed when this option is invoked. If the option is not invoked, draft mode is the default.
–font [**a**|**b**|**c**|**d**|**e**|**g**|**h**]	Supports one of the character sets specified in the *Using Your Aegis Environment.* The value **a, b, c, d, e,** or **g** is used to specify the national character set as outlined in the above manual. In addition, you must specify **h** to switch back to the standard character set (This is useful if you run **prf** in interactive mode). You can also add "font x" to your **prf.db** startup file. The default is **h**, the standard character set.

Options Applying to Plot Files

–res[**olution**] *n*	Output plot resolution in dots per inch. If you specify a resolution not available on the particular printer, **prsvr** prints the file at the closest available resolution. The default resolution is specified in the **prsvr** configuration file.
–white[**_space**] *n*	The amount of white space (in inches) to appear between multiple plots in one file. The default is three inches.
–bw[**_rev**] [**on**|**off**]	Enable/disable black and white reversal for bitmaps. If no argument is specified, **on** is assumed. If the option is not invoked, black/white reversal is disabled.
-magn[**ification**] *n*	Specify bitmap magnification value. *n* is an integer in the range -1 to 16. The values have the following meanings: -1 Selects auto-scaling to magnify the bitmap to fill the available page space. 0 Selects one-to-one scaling between the display and the printer for GMF bitmaps. (For GPR bitmaps, this translates to magnification 1.)

1-16 Selects the magnification indicated by value. Where 1 equals 1-to-1, 2 equals 2x, etc.

Default if omitted: *n* is 0

Options Applying to PostScript* Printers

The following options apply only for files sent to printers that contain the PostScript interpreter, such as the Domain/Laser-26 and APPLE LaserWriter* printers.

–post[script] [on|off]

Enable/disable PostScript interpretation. When enabled, the data is passed through the PostScript interpreter. When disabled, the data is printed as text, plot, or transparent data. If the option is not invoked, PostScript interpretation is disabled.

The default is **on**.

–orient[ation] {port[rait]|land[scape]}

Select the page orientation. **portrait** specifies that the text or x-axis of the bitmap is printed parallel to the short edge of the paper. **landscape** specifies that the text or x-axis of the bitmap is printed parallel to the long edge of the paper and perpendicular to the short leading edge.

The default is **portrait**.

Information Request Options

–check [**–pr** *printer_name*]

Checks for the existence of the specified printer. If the printer does not exist or is unavailable, an error message is returned.

–list_printers Lists the names and status of all printers currently attached to the network.

–list_sites Lists the names of all print managers currently in the network.

–sig_printer *printer_name* {**-abort**|**-sus[pend]**|**cont[inue]**}

Signals the printer to abort, suspend, or continue an active print job.

–pre10 Allows you to queue print requests to a pre-SR10 print server.

COMMANDS

Once **prf** has been invoked in interactive mode (see **–inter** above), it accepts the following interactive commands at the "prf> " prompt (in addition to the options already discussed).

p[rint] [*print_file_pathname*] [*options*]

Queue the specified file for printing.

q[uit]	Quit interactive mode and return to the shell.
sh[ell]	Create a shell command line. This command allows you to issue shell commands without leaving **prf** interactive mode. When you finish entering shell commands, type CTRL/Z. This returns you to **prf** interactive mode. Your previous **prf** option settings remain undisturbed by the intervening shell commands.
init[ialize]	Reset **prf** parameters to their default values.
r[ead] [*printer*]	List queue entries for the specified printer. If *printer* is omitted, the contents of the queue (determined by the current setting of **–pr**) are listed.
wd [*pathname*]	Execute the shell command **wd** (working_directory) to set or display the working directory.
get *option*	Display the value of the **prf** option specified. Use this command to show the settings of the various **prf** parameters.
can[cel] [*job_id*]	Cancel printing of the specified file at the current printer. Note that you must specify the job ID assigned by **prmgr** when the file is queued. Use the **read** command to display the names and job IDs of currently queued files. This command affects jobs in the print queue; it does not cancel a job being printed. To halt a job being printed, use **–pr_sig** with abort specified.

EXAMPLES

The following example, queues the file named **mary** for printing and forces FORTRAN carriage returns:

```
$ prf mary -ftn
"//node1/my_dir/mary" queued for printing.
$
```

The following example queues the file named **filex** to the printer queue on the node named **//tape**:

```
$ prf filex -s //tape
  "//node1/my_dir/test_file.pas" queued for printing at site //tap
$
```

This example shows the types of commands that might appear in the default **prf** configuration file **~/user_data/startup.prf**:

```
pr ge
site //rye
foot %/my_file/&
```

The following example shows a sample interactive session:

```
$ prf -inter
prf> get pr
pr = p
prf> -pr cx
prf> get pr
pr = cx
prf> -pitch 20
prf> print test_file.pas
"//node1/my_dir/test_file.pas" queued for printing.
prf> q
$
```

This example illustrated running **prf** from an icon. To run **prf** interactively in a process devoted to it, insert the following command in the start-up file that you use to start the DM:

```
cp -i -c 'P' /com/prf -inter -n print_file
```

The above command creates a **prf** process and turns its window into an icon using the print icon character in (/sys/**dm/fonts/icons**). Issue the DM command **icon** to change the icon window into its full-size format.

NOTES

APPLE and LaserWriter are registered trademarks of Apple Computer, Inc.
Printronix is a trademark of Printronix, Inc.
Spinwriter is a registered trademark of NEC, Inc.
IMAGEN is a registered trademark of IMAGEN Corp.
GENICOM is a registered trademark of GENDICOM Corp.
Versatec is a trademark of Versatec, Inc.
PostScript is a registered trademark of Adobe Systems, Inc.

SEE ALSO

More information is available. Type

help prfd	For information about the menu-based **prf** command
help printer	For general information about printers supported in a Domain/OS network
help prsvr	For details about the print server
help prsvr/config	For an explanation of the **prsvr** configuration file and its directives, including their default values
help prmgr	For details about the print manager

NAME

prmgr – start the print manager

SYNOPSIS

/sys/**hardcopy**/**prmgr** **–cfg** *configuration_filename* **–n** *process_name*

DESCRIPTION

Print managers coordinate user print requests generated by the **prf** command and control one or more print servers. Print servers are bound to the print managers in the print server configuration file.

When the print manager receives print requests, it checks to ensure that the requested printer exists. If it does not, the job is rejected. If the printer exists, the print manager notifies an appropriate print server. The print server processes the print job and informs the print manager of the ongoing status of the print job.

Starting the Print Manager

Print managers are started with the **prmgr** command. You can start the print manager from any node on which the **llbd** (NCS Local Location Broker daemon) is running on the print manager node. The print manager can be started as either a foreground or background process.

Print Manager Configuration File

As an option to the command, you supply the print manager configuration file name. The print manager configuration file defines the manager's spooling node and logical name and, if appropriate, invokes a print daemon that allows printing of pre-SR10 print jobs.

The print manager configuration file contains three items:

- The print manager logical name, which should identify the type and location of the printers serviced by the print manager
- The print manager's spooling node, a node that includes a /sys/**print** directory (not just a link to that directory)
- An optional switch, pre10q, which allows the SR10 print environment to accept pre-SR10 print jobs

You must define a configuration file for each print manager. A sample configuration file defining a spooling node named //**flash** and a print manager named **r&d** is shown below:

```
spool_node = //flash
prmgr_name = r&d1
pre10q
```

ARGUMENTS

–cfg *configuration_filename*	The the name of the print manager configuration file. **prmgr** looks for the configuration file in the current working directory or specified pathname.
–n *process_name*	The name assigned to the print manager process. We recommend that you use the logical name specified in the configuration file.

SEE ALSO

More information is available. Type

help prf	For information about printing files
help prfd	For information about the menu-based **prf** command
help printer	For general information about printers supported in a Domain/OS network
help prsvr	For information about the print server
help prsvr/config	For information about the print server configuration file

NAME

probenet – probe network and display error statistics

SYNOPSIS

/etc/**probenet** [*options*]

DESCRIPTION

This command broadcasts packets to the diagnostic socket in all nodes, then requests error counts indicating the status the broadcast was received with. It compiles counts from every node in the topology list and reports them to standard output.

OPTIONS

Use one of the following three options to specify the list of nodes to display:

–a (default)	Probe all nodes responding to a /**com/lcnode** command. If the network is completely corrupted so that messages cannot make a complete pass, use one of the other two options to specify precisely which nodes to test.
–t *pathname*	Probe the nodes listed in the topology file indicated. The file must contain one hexadecimal node ID per line. Any text following a space after the node ID is ignored. You can insert comment lines if they are prefixed with a "#" or "{".
–n *node_id* ...	Probe the node(s) specified by the indicated hexadecimal node ID(s). A good choice of nodes to test is a set evenly spaced around the network.

Use the following options to specify which test to run:

–s *n* (default)	Specify the total number of packets to be sent to each node. The default number of packets is 10. If 0 is specified for *n*, no test messages are sent, but statistics from each node are collected.
–r [*n*]	Repeat the **probenet** cycle every *n* seconds. If *n* is omitted, the cycle is repeated every 10 seconds. When you press <RETURN> at the input window, the send cycle is terminated immediately and the statistics are gathered and reported.

Use the following options to specify which packets are sent:

–d *data_file*	Specify that the packets are taken out of the specified data file instead of the standard built-in data pattern.
–len *n* (default)	Specify the length (in bytes) of the data portion of the test packet, in bytes. The default length is 1024 bytes.

Use the following options to control the level of detail in the statistics report.

–l	Print long (detailed) error counts if there were any errors (that is, at least one transmit error (**xmit errs**) or receive error (**rcv errs**).
–err	Print header for each test, but statistics only for nodes which returned errors (**xmit** and/or **rcv** errs).

–mon *fail_lim*

Print header for each pass, but statistics only on passes whose total failure count equal or exceed the *fail_lim* value.

–sens *threshold*

Open a window pane and select some output lines to append to this pane. The nodes selected are those whose error count exceed a five-node running average error count by the specified threshold value. Also, all nodes with modem errors are appended to this pane. The use of this secondary output is to do some data reduction and pick the nodes at or near points of data corruption in the network. The window pane is also stored in a named pad file called **probenet.pane**.

EXAMPLES

```
1. $ /etc/probenet {Probe entire network once.  No errors detected.}
There are  5 nodes in the test.
Broadcasting 10 1024-byte packets . 85/02/20 21:16:52  # failures =  0
Last Biph hardware failure detected by node 676 on 85/02/20 at 19:15

                                      MODEM
 NODE  NAME          ATTEMPT  ERRS    ERRS     BIPH    ESB  TOKENS= 0
 ----  ------------  ------ ------  ------    ----    ---  ------
  584  *diskless         10      0       0       0      0       0  Self
   21  OS                10      0       0       0      0       0
  AEF  BS                10      0       0       0      0       0
   4A  HUBRIS            10      0       0       0      0       0
 3536  *diskless         10      0       0       0      0       0
```

```
2.  $ probenet -t node_list -s 14400 -r 3600 -d data_e3

   { Probes network and displays nodes specified in file "node_list".
     This node broadcasts 14400 packets in 3600 seconds, that is, four
     packets per second. The packet data comes out of file "data_e3".}

There are      5 nodes in the test.
Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.
    85/02/20 21:58:19  # failures =  100
Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

                                      MODEM
 NODE  NAME          ATTEMPT  ERRS    ERRS     BIPH    ESB  TOKENS= 3
 ----  ------------  ------ ------  ------    ----    ---  ------
 1967  GTX            14386      0       0       0      0       1  Self
 15F5  SWI            14386      0       0       0      0       0
 2255  BIRDIE         14384      0       0       0      0       1
```

```
  3FD  FLASH          14386     3      3      3      0      0
 2B69  STANG          14385     3      0      0      0      1

Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.
    85/02/20 21:58:41  # failures =  100
Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

                                        MODEM
 NODE  NAME          ATTEMPT  ERRS    ERRS    BIPH    ESB  TOKENS= 3
 ----  ------------   ------ ------ ------    ----    ---  ------
 1967  GTX            14383     0      0      0      0      1  Self
 15F5  SWI            14383     0      0      0      0      0
 2255  BIRDIE         14381     0      0      0      0      1
  3FD  FLASH          14383     4      4      4      0      0
 2B69  STANG          14382     4      0      0      0      1

{ Above example shows a problem between node 3FD and its predecessor
  in the network. }
```

SEE ALSO

More information is available. Type

help lcnode	For details about determining which nodes are currently connected to the network

NAME

prsvr – start the print server

SYNOPSIS

/sys/**hardcopy**/**prsvr** [*config_file_name*] [**–n** *name]*

DESCRIPTION

prsvr is the command that starts the print server process, which handles the processing of files submitted by the print manager for printing. Print servers determine the print parameters and send print requests to a selected printer. The are bound to the print managers in the print-server configuration file.

Print servers are started (typically as a background task) by executing the **prsvr** command. **–n** specifies the name of the printer configuration file that defines the printer and its print parameters. You must execute this command whenever you start or restart the node connected to the printer. To avoid losing print files already queued, do not execute the **prsvr** command at nodes without an attached printer.

ARGUMENTS

config_file_name	The name of the print-server configuration file.

OPTIONS

–n *name*	The name of the print-server process. The default is **print_server**.*printername*. *printername* is the device name specified in the print server configuration file.

SEE ALSO

More information is available. Type

help prf	For information about printing files
help prfd	For information about the menu-based **prf** command
help printer	For general information about printers supported in a Domain/OS network
help prsvr/config	For an explanation of the **prsvr** configuration file and its directives, including their default values
help prmgr	For details about the print manager

NAME

prsvr/config – Print Server Configuration Directives

DESCRIPTION

Configuration directives for the print server are read from a file when the print server is started. The name of this file is specified on the command line, following the **prsvr** command.

Note that some of the options and arguments below now can be overridden by user options to the **prf** command. Additionally, some new options make older options obsolete. The older options still are functional, and are the defaults in some circumstances. See individual descriptions for complete explanations.

The following configuration directives are recognized:

Directive	Meaning
bottom_margin [*n*]	Number of lines to skip at the bottom of the page. This option can be used only if the **prf** command does not contain a margin specification. Default if omitted: 4
collate_copies [on|off]	Enables/disables page collation. Valid on the Domain/Laser-26 or any other PostScript printer. Default if omitted: off

cpi [*n*]	The number of characters per inch (the pitch) printed by the current printer. Use this option if you use a non-default pitch for a printer. Default values are as follows:
	
	`IMAGEN      12`
	`SPIN        12`
	`PRINTR      10`
	`VER         12`
	`GE          12`
	`LASERWRITER 12`
	`LASER26     12`

ddf_name [*pathname*]	Specify a device descriptor file for a MULTIBUS* Controller. This option is useful for driving a printer with a MULTIBUS card. Apollo-supplied printers which may use this option are the IMAGEN and Versatec printers. Default if omitted: **/dev/versatec**

device

Specify the Domain supported printer types or user supplied device drivers which print files. The following are valid devices:

Default if omitted: spinwriter

```
[
spinwriter |
printronix |
versatec |
ge |
imagen |
laserwriter |
laser26 |
user1 |
user2 |
user3 |
user4
]
```

file_banners

Specify whether you want a banner page preceding each file, following each file, or suppressed.

Default if omitted: first

```
[
  first |
  first last |
  off
]
```

form_feeds [*n*]

Specify the number of pages to form feed between jobs.

Default if omitted: 1

interface

Specify the hardware interface used by an IMAGEN printer. The hardware interfaces are:
serial – SIO line;
versatec – Versatec IKON 10071 and 10085 boards;
multibus – user-supplied MULTIBUS controller.

```
[
serial |
versatec |
multibus
]
```

lpi [*n*]

Specify the number of lines per inch printed by the current printer. Use this option if you don't use the default number of lines per inch on a printer.

```
imagen      6
spin        6
printr      6
vers        6
GE          6
laserwriter 7.2
laser26     7.2
```

logo [string|none]

Specify a character string to be printed on the banner page.

Default if omitted: none

model_number[*xxx*]

Currently, this option applies to printers with multiple model numbers, i.e.:

```
imagen 8/300 for the CX
LBP-10       for the LBP-10
IP3          for the IP3 controller
versatec     V8236
             V8224
             V8244
             V8272
             V80
```

pageno_column [*n*]

Specify column in which page number is printed in the header. May be overridden by user options to the **prf** command.

Default if omitted: 90

page_headers [on|off]

On prints a one line header at the top of each page, containing the filename and page number. May be overridden by user options to the **prf** command.

Default if omitted: on

page_length [*n*]

Length, in lines, of page. May be overridden by user options to **prf** command. This is the default option if **print_length** is not specified.

Default if omitted: 66

page_reversal [on|off]

The normal page printing order [off] is reverse collation (last page printed first). On causes **prsvr** to reverse this order of pages as they are printed on the Domain/Laser-26 or any other PostScript printer (first page printed first).

Default if omitted: off

page_width [*n*]

Width, in characters, of page. If the input line length exceeds the specified page width, the excess charaters are truncated and a warning message appears, listing the number of truncated lines. May be overridden by user options to the **prf** command. This is the default option if **print_width** is not specified.

Default if omitted: 132

paper_size *size*

Set the default paper size used by the Domain/Laser-26 printer. This is used if **–paper_size** option is omitted from the **prf** command line. This also overides the default paper tray selection if the default tray does not contain the specified paper size. This directive may be used instead of the **print_length** and **print_width** directives. Specify *size* as 'a', 'b', etc. using the chart below.

Size	Width	Length
a	8.5"	11"
b	11"	17"
legal	8.5"	14"
statement	5.5"	8.5"
a3	297 mm	420 mm
a4	210 mm	297 mm
a5	137 mm	210 mm
b4	257 mm	364 mm
b5	182 mm	257 mm

plot_mode [on|off]

Specify whether the device will accept plot files.

Default if omitted: off

print_length [*inches*]	Specify in inches the length of paper which can be used for printing, i.e., the length of paper minus any limits set by the printer's physical capabilities, including the margins you physically set on the printer. For example, IMAGEN printers will take paper size 8.5 x 11 inches but can only print on an area size 8.00 by 10.8 inches. Use this command instead of **page_length** to specify page format.
print_width [*inches*]	Specify in inches the width of paper which can be used for printing, i.e., the width of paper minus any limits set by the printer's physical capabilities, including the margins you physically set on the printer. For example, if you use Spinwriter operator settings, the **print_width** option must reflect those settings. Use this command instead of **page_width** to specify page layout.
printer_name [*string*]	Specify printer name used in the **–pr** option of the **prf** command; useful when several printers are attached to a single node. Default if omitted: p

resolution [*n*]	Specify the resolution of the printer in dots per inch. If you use the IMAGEN cx, set the resolution at 300 dots per inch. You may also choose a resolution of 144 dots per inch for the GE printer if the firmware configuration is 403277 or greater. May be overridden by user options to the **prf** command. The default values are:
	
	`cx      300`
	`ge       72`

sio_line [*n*]	Specify SIO line to which printer or user device is attached; not meaningful for Printronix or Versatec printers. If SIO line 3 does not exist (as on DN3xx systems), the default line is 2. Default if omitted: 1
speed [*n*]	Specify baud rate for SIO line; not meaningful for Printronix or Versatec printers.
top_margin [*n*]	Specify number of lines to skip at the top of the page. This option can be used only if the **prf** command does not contain a margin specification.

EXAMPLES

Sample configuration files might look like these:

Printronix Printer:

```
print_width    13.0
print_length   11.0
bottom_margin  4
pageno_column  90
resolution     66
form_feeds     1
file_banners   on
page_headers   off
plot_mode      on
printer_name   p
device         printronix
logo           <none>
```

IMAGEN cx Printer:

```
print_width      8.0
print_length    10.8
page_headers    off
pageno_column   72
file_banners    on
device          imagen
plot_mode       on
interface       serial
speed           19200
printer_name    cx
sio_line        1
resolution      300
logo            <none>
```

SEE ALSO

prsvr	For details about starting the print server
prf	For details about queueing files to a printer

NAME

rexecd – remote execution server

SYNOPSIS

/etc/rexecd

DESCRIPTION

rexecd is the server for the **rexec** routine. The server provides remote execution facilities with authentication based on user names and passwords.

The **rexecd** server listens for service requests at the port indicated in the ''exec'' service specification; see **services**. When a service request is received the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

```
username too long
```

The name is longer than 16 characters.

```
password too long
```

The password is longer than 16 characters.

```
command too long
```

The command line passed exceeds the size of the argument last (as configured into the system).

```
Login incorrect
```

No password file entry for the user name existed.

```
Password incorrect
```

The wrong was password supplied.

```
No remote directory
```

The **chdir** command to the home directory failed.

```
Try again
```

A **fork** by the server failed.

```
<shellname>: ...
```

The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

BUGS

Indicating ''Login incorrect'' as opposed to ''Password incorrect'' is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.

SEE ALSO

rexec

NAME

/etc/rgy_admin – registry server administrative tool

SYNOPSIS

/etc/rgy_admin

DESCRIPTION

The **rgy_admin** tool administers registry servers. It can view or modify the registry replica list, reinitialize replicas, delete replicas, stop servers, and change the registry master site.

Note that **rgy_admin** cannot add, delete, or modify data entries contained in the registry database, such as names and accounts; use **edrgy** to perform these tasks. To create a registry replica or to restart a server, use **rgyd**, the registry daemon.

Once invoked, **rgy_admin** enters an interactive mode in which it accepts the commands described in the next section.

COMMANDS

Most **rgy_admin** commands operate on a default host. You use the **set** command to establish the default host, which is remembered until changed by another **set**. In the following command descriptions, we identify the default host as *default_host*. If a command operates on a host other than the default, we identify this host as *other_host*.

Several of the **rgy_admin** commands require you to set the default host to the master registry site.

The host name you supply as a *default_host* or *other_host* takes the form *family*:*host* or *host*. The only currently supported *family* is **dds**, the Domain protocol family. You can specify a host in this family by its entry directory or by its network address. For example, **dds://clara**, **//clara**, **dds:#1234.abcd**, and **#1234.abcd** are all acceptable host names.

become [–master] [–slave] [–ro | –wr]

The **–master** option causes the replica at *default_host* (which must be a slave replica) to become the master. This operation can cause updates to be lost; the **change_master** command is the preferred means of designating a different master replica.

The **–slave** option causes the replica at *default_host* (which must be the master replica) to become a slave. This operation can cause updates to be lost; the **change_master** command is the preferred means of designating a different master replica.

The **–ro** option makes the replica list read-only. The *default_host* must be set to the master registry site.

The **–wr** option makes the replica list writable. The *default_host* must be set to the master registry site.

change_master –to *other_host*
: Change the master replica of the registry from *default_host* to *other_host*. The *default_host* must be set to the master registry site.

  The current master server copies its database to the replica at *other_host*, becomes a slave, then tells the replica at *other_host* to become the master.

delrep *other_host* [**–force**]
: Delete the registry replica at *other_host*. The *default_host* must be set to the master registry site.

  The master server marks the replica at *other_host* as deleted and propagates the deletion to all other replicas on its list. When it has actually delivered the delete request to the replica at *other_host*, the master server removes that replica from its own replica list.

  The **–force** option causes a more drastic delete. It deletes *other_host* from the replica list at the master registry, which then propagates the delete request to the replicas at the hosts that remain on its list. Since this operation never communicates with the deleted replica, you should use **–force** only when the replica has died irrecoverably. If you use **–force** while the replica at *other_host* is still running, you should then **reset** the deleted replica.

help List the **rgy_admin** commands and show their allowed abbreviations.

info Get status information about the replica at *default_host*.

initrep *other_host*
: Reinitialize the registry server at *other_host*. The *default_host* must be set to the master registry site. The *other_host* must be a slave site.

  The master registry copies its entire database (or that of another up-to-date replica) to the replica at *other_host*.

lrep [**–state**] [**–na**]
: List the registry replica sites as stored in the replica list at *default_host*.

  The **–state** option shows the current state and update time on each host.

  The **–na** option shows the network address of each host.

monitor [**–r** *m*]
: Periodically list the registry replica sites as stored in the replica list at *default_host* and show the current state and update time at each site.

  The **–r** option causes the sites to be listed every *m* minutes. If you omit this option, the period is 15 minutes.

quit Quit the **rgy_admin** session.

reprep *other_host*

Replace the network address for *other_host* in the registry replica list. The *default_host* must be set to the master registry site.

The master replica propagates the new network address for *other_host* to all other registry replica lists. Use this command only if a replica site's network number changes.

reset *other_host*

Reset the registry replica at *other_host*. The registry server at *other_host* deletes its copy of the registry and stops running. This command does not delete *other_host* from any replica lists.

set [**–h** *host_name* | **–m**]

Set the default host. Subsequent commands that do not specify a host will go to this host.

The **–h** option specifies a replica to use as the default.

The **–m** option causes the master replica to be the default.

With no options, **set** locates a registry replica and sets it as the default.

site [*host_name*]

If *host_name* is specified, the command sets the default host.

If *host_name* is not specified, the command gets status information about *default_host*.

state –in_maintenance | –not_in_maintenance

Put the master registry server into maintenance state or take it out of maintenance state. The *default_host* must be set to the master registry site.

With the **–in_maintenance** flag, **state** causes the master registry server to save its database onto disk and refuse any updates.

With the **–not_in_maintenance** flag, **state** causes the master registry server to reload its database from disk, return to its normal "in service" state, and (if its database and/or replica list are writable) start accepting updates.

stop Stop the registry server that is running at *default_host*.

EXAMPLES

Start **rgy_admin**, list the replicas and their states, then set the default host to the master replica:

```
$ /etc/rgy_admin
Default object: rgy  default host: dds://george
State: in service  slave
rgy_admin: lrep -st
  (master) dds://martha  state: in service  1987/11/16.12:46:59
           dds://george  state: in service  1987/11/16.12:46:59
           dds://thomas  state: in service  1987/11/16.12:46:59
rgy_admin: set -m
          Default object: rgy  default host: dds://martha
          State: in service  master  replica list is writeable
```

NAME

rgy_create – registry creation utility

SYNOPSIS

rgy_create

DESCRIPTION

The **rgy_create** tool creates a new registry database on the local node, initialized with reserved names and accounts. It ordinarily should be run only once at a site. Replicas of the database are created by running **rgyd** with the **–create** option.

You must be root to invoke **rgy_create**.

Note that to convert a pre-SR10 registry to SR10 format, you should run only the **cvtrgy** tool, and you will never need to use **rgy_create**.

NAME

rgy_merge – merge registry database

SYNOPSIS

rgy_merge –from *//site* [{ **–merge** | **–compare** } **–verbose**]

DESCRIPTION

The **rgy_merge** utility merges the contents of two registry databases, a source database and a target database. You typically use it when you are joining two networks that have been operating separately and you want to combine their registry databases.

You must invoke **rgy_merge** while logged in as root at the master registry node for the target registry. Use the required **–from** *//site* argument to specify the master registry node for the source registry. The merge takes less time if the source database is smaller than the target database.

After you invoke **rgy_merge**, the tool prompts you to "login" with an account that owns the source registry.

Without a **–merge** or **–compare** option, **rgy_merge** attempts to add each entry in the source database to a copy of the target database and reports any conflicts in names or UNIX numbers. If there are no conflicts or errors, the tools asks whether you want to actually update the target database. If you respond affirmatively, it performs the merge on the target database and makes all replicas of the source registry slave replicas of the target registry.

If you specify **–merge**, **rgy_merge** performs the merge without querying you, provided there are no conflicts or errors. If you specify **–compare**, **rgy_merge** only checks for conflicts and does not perform a merge even if there are none.

The **–verbose** option causes **rgy_merge** to generate a verbose transcript of its activity.

NAME

rgyd – network registry server

SYNOPSIS

/etc/**rgyd** [**–create**|**–recreate**|**–restore_master**]

DESCRIPTION

rgyd is the network registry daemon. It manages all access to the network registry database. You must be the super-user to invoke **rgyd**.

The daemon can be replicated, so that several copies of the database exist on a network or an internet, each managed by a **rgyd** process. Only one registry daemon, the master, can accept operations that change the database (such as adding an account). If the daemon is replicated, the other replicas are slaves, which accept only lookup operations (such as validating a login attempt).

A Local Location Broker daemon (**llbd**) must be running on the local node when **rgyd** is started. Typically, both daemons are started at boot time from /etc/**rc**. The server will place itself in the background when it is ready to service requests.

OPTIONS

–create	Create a replica of the network registry. This option creates a copy of the registry database and starts a slave server process. You use **–create** only the first time you start a slave server process on a node. When you restart the daemon, you do not need any options at all. To create the master replica, use either **cvtrgy** (if you are converting an SR9 registry to SR10 format) or **rgy_create** (if you are creating a new SR10 registry).
–recreate	Recreate a slave replica. You should use this option only if a slave's copy of the database has been irreparably corrupted. It destroys the existing database and creates a new one.
–restore_master	Restart a master server and reinitialize all slave replicas. You should use this option only to recover from a catastrophic failure of the master node, (for example, if the database has been corrupted and then restored from a backup tape).

EXAMPLES

All of the commands shown in these examples must be run by root.

1. Start the master replica of the registry after you have created the master database via **rgy_create** or **cvtrgy**:

```
$ /etc/server -p /etc/rgyd
```

2. Start a slave replica of the registry.

```
$ /etc/server -p /etc/rgyd -create
```

3. Restart an existing replica (master or slave) of the registry.

   ```
   $ /etc/server –p /etc/rgyd
   ```

4. Restart an existing replica of the registry on the remote host //yak.

   ```
   $ /etc/crp –on //yak –cps –n rgyd //yak/etc/rgyd
   ```

SEE ALSO

cvtrgy, rgy_admin, rgy_create, rgy_merge

NAME

rlogind – remote login server

SYNOPSIS

/etc/rlogind [**–d**]

DESCRIPTION

rlogind is the server for the **rlogin**(1C) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

rlogind listens for service requests at the port indicated in the ''login'' service specification; see **services**. When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server checks the client's source address and requests the corresponding host name (see **gethostbyaddr**, **hosts** and **named**. If the hostname cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, **rlogind** allocates a pseudoterminal (see **pty**), and manipulates file descriptors so that the slave half of the pseudoterminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r** option. The login process then proceeds with the authentication process as described in **rshd**, but if automatic authentication fails, it reprompts the user to log in as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudoterminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in **pty** is invoked to provide CTRL/S and CTRL/Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, ''TERM''; see **environ.** The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudoterminal.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

```
Try again.
```

A **fork** by the server failed.

```
/bin/sh: ...
```

The user's login shell could not be started.

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ''open'' environment.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol should be used.

NAME

route – manually manipulate the routing tables

SYNOPSIS

/etc/**route** **[–f]** **[–n]** [*command args*]

DESCRIPTION

route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, **routed**, should tend to this task.

route accepts three commands: **add**, to add a route; **addp**, to add a priority route; and **delete**, to delete a route.

The **addp** command adds a priority route. The TCP/IP server process will use priority routes before default routes or routes established by **routed**. A route added with **addp** will appear first in the gateway table (displayed with the Aegis command **netstat –r**). You can only add priority routes with **addp**. Routes added manually by **route** cannot be deleted by **routed**.

COMMAND SYNTAX

All commands have the following syntax:

/etc/**route** *command* [**net** | **host**] *destination gateway* [*metric*]

where *destination* is the destination host or network, *gateway* is the next-hop gateway to which packets should be addressed, and *metric* is a count indicating the number of hops to the *destination*. The metric is required for **add** and **addp** commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission.

Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. The optional keywords **net** and **host** force the destination to be interpreted as a network or a host, respectively. If the *destination* has a "local address part" of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

If the route is to a destination connected through a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname**. If this lookup fails, **getnetbyname** is then used to interpret the name as that of a network.

You can add a default route as follows:

/etc/**route** *add default gateway_name* [*non-zero metric*]

TCP/IP software will use the default route when other routes occuring earlier in the routing table have failed, or when there are no other possible routes.

route uses a raw socket and the SIOCADDRT and SIOCDELRT **ioctl**'s(2) to do its work. As such, only the super-user may modify the routing tables.

OPTIONS

–f ''Flush'' the routing tables of all gateway entries. Using this option in conjunction with one of the commands described above flushes the tables prior to the command's application.

–n Suppress printing symbolic host and network names when reporting actions.

DIAGNOSTICS

```
add [ host | network ] %s: gateway %s flags %x
```

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl**(2) call. If the gateway address used was not the primary address of the gateway (the first one returned by **gethostbyname**), the gateway address is printed numerically as well as symbolically.

```
delete [ host | network ] %s: gateway %s flags %x
```

As above, but when deleting an entry.

```
%s %s done
```

When the **–f** flag is specified, each routing table entry deleted is indicated with a message of this form.

```
Network is unreachable
```

An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

```
not in table
```

A delete operation was attempted for an entry that wasn't present in the tables.

```
routing table overflow
```

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

routed;
Configuring and Managing TCP/IP.

NAME

routed – network routing daemon

SYNOPSIS

/etc/routed [**–g**] [**–s**] [**–q**] [**–t**] [**–n**] [**–f**] [**–h**] [*logfile*]

DESCRIPTION

The **routed** daemon is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries. It uses a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation **routed** listens on the **udp** socket for the **route** service (see **services**) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **routed** is started, it uses the SIOCGIFCONF **ioctl** to find those directly connected interfaces configured into the system and marked ''up'' (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host will forward packets between networks. **routed** then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, **routed** formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a ''hop count'' metric (a count of 16, or greater, is considered ''infinite''). The metric associated with each route returned provides a metric *relative to the sender*.

Response packets received by **routed** are used to update the routing tables if one of the following conditions is satisfied:

1) No routing table entry exists for the destination network or host, and the metric indicates the destination is ''reachable'' (i.e. the ''hop count'' is not infinite).

2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; to decide this, the metric of the new route is compared against the one stored in the table.

When an update is applied, **routed** records the change in its internal tables. The change is reflected in the next *response* packet sent.

In addition to processing incoming packets, **routed** also checks the routing table entries periodically. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to ensure that the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

OPTIONS

routed supports several options:

–g This flag is used on internetwork routers to offer a route to the "default" destination. This option is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

–s Forces **routed** to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

–q This option is the opposite of the **–s** option.

–t If the **–t** option is specified, all packets sent or received are printed on the standard output. In addition, **routed** will not divorce itself from the controlling terminal, so that interrupts from the keyboard will kill the process.

–d Not supported by Domain/OS Aegis.

Domain/OS Aegis EXTENSIONS

–n Directs **routed** not to install changes into the local routing table. However, the **routed** process continues to receive broadcasts from other **routed** processes. The **–n** option is used for debugging purposes.

–f Directs **routed** at startup to "flush" (purge) all routes from the local routing table, except routes added manually with **/etc/route**.

–h Exit, if not supplier, when routing table is stable. Use this switch on hosts only, not on gateways.

Any other argument supplied is interpreted as the name of the file in which **routed**'s actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, **routed** supports the notion of distant passive and active gateways. When **routed** is started up, it reads the file /etc/**gateways** to find gateways that may not be located using only information from the SIOGIFCONF **ioctl**. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a **routed** process running on the machine).

Passive gateways are maintained in the routing tables forever, and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted. External gateways are also passive, but they are not placed in the routing table nor are they included in routing updates The function of external entries is to inform **routed** that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The /etc/**gateways** file is comprised of a series of lines, each in the following format:

< **net** | **host** > *name1 gateway name2 metric value* < **passive** | **active** | **external** >

The **net** or **host** keyword indicates if the route is to a network or specific host.

Name1	The name of the destination network or host. This may be a symbolic name located in /etc/**networks** or /etc/**hosts** (or, if started after **named**, known to the name server), or an Internet address specified in ''dot'' notation; see **inet**.
Name2	The name or address of the gateway to which messages should be forwarded.
Value	A metric indicating the hop count to the destination host or network.

One of the keywords **passive**, **active** or **external** indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the **routed** protocol.

Internetwork routers that are directly attached to the ARPANET or Milnet should use the Exterior Gateway Protocol (EGP) to gather routing information rather then using a static routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. Sites needing assistance with such configurations should contact the Computer Systems Research Group at Berkeley.

For a node to run **routed**, it must be correctly configured to run Aegis TCP/IP. See *Configuring and Managing TCP/IP* for more information about **routed**.

NOTES

The **routed** daemon is normally started on a node at boot time, from the /etc/rc.local file. We recommend that you run **routed** on each gateway to dynamically update the gateway's routing tables. You can also run **routed** on hosts so they receive the latest routing information.

FILES

/etc/gateways	for distant gateways

BUGS

The kernel's routing tables may not correspond to those of **routed** when redirects change or add routes. The only remedy for this is to place the routing process in the kernel.

routed does not incorporate other routing protocols, such as Xerox NS and EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

routed does not currently listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).

SEE ALSO

udp, htable, route, rc;
Configuring and Managing TCP/IP;
"Internet Transport Protocols", XSIS 028112, Xerox System Integration Standard.

NAME

rshd – remote shell server

SYNOPSIS

/etc/rshd

DESCRIPTION

rshd is the server for the **rcmd** routine and, consequently, for the **rsh**(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts.

rshd listens for service requests at the port indicated in the ''cmd'' service specification; see **services**. When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address and requests the corresponding host name (see **gethostbyaddr**, **hosts** and **named**). If the hostname cannot be determined, the dot-notation representation of the host address is used.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client 's machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server 's machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) **rshd** then validates the user according to the following steps. The local (server-end) user name is looked up in the password file and a **chdir** is performed to the user's home directory. If either the lookup or **chdir** fail, the connection is terminated. If the user is not the super-user, (user id 0), the file **/etc/hosts.equiv** is consulted for a list of hosts considered ''equivalent''. If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection

is terminated.

9) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the execution of the login shell).

```
locuser too long
```

The name of the user on the client's machine is longer than 16 characters.

```
remuser too long
```

The name of the user on the remote machine is longer than 16 characters.

```
command too long
```

The command line passed exceeds the size of the argument list (as configured into the system).

```
Login incorrect.
```

No password file entry for the user name existed.

```
No remote directory.
```

The **chdir** command to the home directory failed.

```
Permission denied.
```

The authentication procedure described above failed.

```
Can't make pipe.
```

The pipe needed for the **stderr**, wasn't created.

```
Try again.
```

A **fork** by the server failed.

```
<shellname>: ...
```

The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ''open'' environment.

SEE ALSO

rsh, rcmd;
Configuring and Managing TCP/IP.

NAME

rtchk – test traffic between adjacent routers

SYNOPSIS

/etc/**rtchk** [*options*]

DESCRIPTION

rtchk performs a simple test to verify that the router is able to pass packets to and from an adjacent router. **rtchk** is for use in a Domain internet.

Use the **–device** option to specify a network controller to test. You must give a device type (for example, RING, IIC) to the device option. The **rtsvc** program, with no command-line options, shows you which network devices your node has.

Older versions of **rtchk** used a different command-line syntax to specify the type of network hardware checked. The old command-line options still work in **rtchk** version 10.1, but are no longer supported.

For more information on **rtchk**, see *Managing Domain Routing and Domain/OS in an Internet*.

OPTIONS

–n *net.node_id*
: Test packet transmission to and from the specified node. The network id *net* must be a network that the router touches. If you use **–n** without **–dev**, you must specify a *net.node_id*. If you use **–n** with **–dev**, you must specify only the *node_id*, without the network number.

–dev[ice] *dev-name [dev-num]*
: Test packet transmission over a specific network device. Use the **rtsvc** program to display the names (used for *dev-name*) and controller numbers (used for *dev-num*) of the network devices attached to your node.

: If you do not also specify a **–n** option, **rtchk** broadcasts its test packets. If the network contains more than one other node, **rtchk** receives more responses to its test packets than expected and prints warning messages. If you specify a **–n** option with the **–dev** option, **rtchk** sends the test packets only to the node you specify.

–s *n*
: Specify the number of test packets to exchange with the other router. If –s is not specified, 10 packets are exchanged.

–dat (default)
: Specify that each test packet carries 1024 bytes of test data.

–nodat
: Omit test data from the test packets.

EXAMPLES

Exchange 1000 test packets with node 4851 on network 3CE02A8. The router must be attached to that network.

```
$ /etc/rtchk –n 3CE02A8.4851 –s 1000
```

Exchange 10 short test packets with the other node attached to the IIC or T1 connection.

```
$ /etc/rtchk –nodat
```

Exchange 100 test packets with the other node on the IIC or T1 network.

```
$ /etc/rtchk –dev iic –s 100
```

Exchange 10 test packets with node 666 on the ring network.

```
$ /etc/rtchk –dev ring –n 666
```

SEE ALSO

More information is available. Type

help rtsvc	For information on listing networks which touch your node

NAME

rtstat – display internet router information

SYNOPSIS

/etc/rtstat [*options*]

DESCRIPTION

rtstat shows the behavior of an internet router at each of its network ports. **rtstat** is used in Domain internets. However, it can provide information about non-routing nodes as well as routing nodes.

For more information on **rtstat**, see *Managing Domain Routing and Domain/OS in an Internet.*

OPTIONS

–dev Report device-specific statistics for each port.

–net [*net_id* ...]
Report counts of references to each network specified. The reference counts for each network are roughly proportional to the number of packets transmitted towards the network, but may be somewhat higher. **–net** with no arguments uses the list of visible networks.

–r [*n*] Repeat every *n* seconds. If *n* is omitted, repeat every 10 seconds.

–n *node_spec* ...
Report statistics for each node in the list.

See **help node_spec** for details about node specification syntax.

If this option is omitted, **rtstat** reports statistics for the local node only.

–desc[ribe] Print a description, several lines long, of each statistic. The description appears only once for each statistic, the first time it is printed with a nonzero value.

EXAMPLES

1. $ **/etc/rtstat**

```
----------------------------------------------------------------
1232.3D9           pkts routed:   110024    queue oflo:        0
                   misrouted:          0    rt too far:       14

     RING          pkts sent:      73278    pkts rcvd:     72434

     IIC           pkts sent:      67830    pkts rcvd:     61077
```

2. $ **/etc/rtstat -net**

```
-------------------------------------------------------------
1232.3D9          pkts routed:  110024    queue oflo:       0
                  misrouted:         0    rt too far:      14

     RING         pkts sent:     73278    pkts rcvd:    72434
                  towards net:    1232    ref cnt:      74540

     IIC          pkts sent:     67830    pkts rcvd:    61077
                  towards net:    1234    ref cnt:      53532
                  towards net:    1231    ref cnt:       9193
                  towards net:    1233    ref cnt:       5105
```

SEE ALSO

More information is available. Type

help netstat	For information about displaying other kinds of node behavior

NAME

rwhod – system status server

SYNOPSIS

/etc/**rwhod** [–w]

DESCRIPTION

The **rwhod** server maintains the database used by the **rwho** and **ruptime** programs. Its operation is predicated on the ability to broadcast messages on a network.

On Apollo networks, **rwhod** has been changed to take advantage of the Domain distributed file system and to reduce contention for the **rwho** directory. Since on Apollo networks, there is no need for every host on the network to maintain its own /usr/spool/**rwho** directory, a change has been made to allow you to specify one host per network who will write the information to a master /**usr/spool/rwho** directory. All the other hosts link to that master directory to access the information but do not write to the directory. Specifically, **rwhod** writes to /**usr/spool/rwho** only if that directory is local to the node, or if the optional –w switch is supplied.

rwhod both produces and consumes system status information. It periodically queries the state of the system and constructs status messages which are broadcast on a network, and it listens for other **rwhod** servers' status messages as well. When it receives a status message from another server, **rwhod** validates it and records it in a file located in the directory /**usr/spool/rwho**, if the directory is physically located on the host or if **rwhod** was started with the –w switch. On hosts where **rwhod** was started without the –w switch and where /**usr/spool/rwho** is a link to a remote host, **rwhod** does not write to the directory.

The **rwho** server transmits and receives messages at the port indicated in the ''rwho'' service specification. The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];/* tty name */
        char    out_name[8];/* user id */
        long    out_time;/* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int        we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order before transmission. The load averages represent averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the **gethostname**(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes an entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at an **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named **whod.hostname** in the directory **/usr/spool/rwho**. These files contain only the most recent message, in the format described above.

OPTIONS

–w Update the files in the **/usr/spool/rwho** directory, even if the directory is not local to the node.

NOTES

Note that **rwhod** will update the files in the **/usr/spool/rwho** directory in two cases:

1. If the **/usr/spool/rwho** directory is physically located on that host.

2. If **rwhod** was started with the –w switch.

To reduce contention for the rwho directory and provide updated information to hosts, we recommend that you

- Physically locate the **/usr/spool/rwho** directory on the network's TCP/IP administrative node. All other hosts on the network should be set up so that **/usr/spool/rwho** is a link to the administrative node.
- If you are running **rwhod** in a Domain internet, you can configure your network in one of two ways:
  1. Configure one node on each subnet to have a local **/usr/spool/rwho** and link all other nodes on that subnet to that node.

  2. Configure all nodes in the internet to link to one master TCP/IP administrative node containing the **/usr/spool/rwho** directory. This enables the **rwho** and **ruptime** utilities to see all nodes in the internet.

In order to make this arrangement work, you must run **rwhod** –w on one node on each subnet. They will then write their subnet's information to the directory on the internet's master TCP/IP administrative node.

You also can run **rwhod** on a gateway to see broadcasts from two networks.

rwhod, like other Aegis daemons, is invoked at boot time by the /**etc**/**rc.local** startup file. See *Configuring and Managing TCP/IP* for more information about **rwhod**.

NOTES

People often interpret the server's dying as a machine in the network going down.

SEE ALSO

rwho, ruptime, rc

NAME

salvol – verify and correct allocation of disk blocks

SYNOPSIS

from shell: /etc/**salvol** [*options*] [*lv_num*]
from mnemonic debugger: **ex salvol**

DESCRIPTION

Each logical volume is divided into disk blocks. **salvol** verifies and, if necessary, corrects the tables that describe the allocation of disk blocks to the files stored on the disk. **salvol** also returns to the free space pool all blocks that are no longer in use: those allocated to temporary files or to deleted portions of permanent files.

salvol can usually restore a disk after a system crash or an improper unmounting of a volume.

If no options are specified on the command line, then **salvol** prompts for all required arguments and options.

lv_num (optional)
: A decimal value for which logical volume number to salvage.

OPTIONS

–a Read all blocks in all files. This option will take longer to run and is useful for finding block header errors or file blocks that cannot be read. This option is not useful for finding multiply allocated blocks or general disk problems.

–c *c*[*cnum*:[*d_num*]
controller type:
c = {w, s, f} (for winchester, storage module or floppy)
cnum = controller number, if specified, must be followed by a ':'
d_num = drive number

This flag must be specified if any command line options are used.

–f Fix error without prompting. (The default is to prompt.)

–h Print help text.

–n Check disk; only salvage if disk needs salvaging. This option is useful in scripts that mount secondary disks at boot time.

–p Polite mode; pause before overflowing screen. (The default is offline.)

–s Show some file statistics at completion.

–t Terminal mode, do not pause during output. (The default is online.)

–v Verify only; do not write anything to disk.

NAME

server – run a server process

SYNOPSIS

/etc/**server** [**–p**] "*command-pathname arg1 arg2 ...*"

DESCRIPTION

The **server** command runs a program with an identity of user "user" and group "server" just as if the command had been started using the Display Manager's **cps** command. It also marks the new process in such a way that the server program will not be terminated by the Display Manager when the user logs out.

In addition to allowing users to start server processes, this command is useful for killing servers that are running with the identity **user.server**. For example,

```
$ /etc/server /bin/kill –9 1532
```

OPTIONS

–p The **–p** option preserves the current SID of the person issuing the command. Otherwise, /etc/**server** sets the SID to **user.server.none** for the command.

NAME

syncids – fix or verify file owners in a file system

SYNOPSIS

/etc/syncids [**–a**] [**–l**] [**–v**] *volume-pathname*

DESCRIPTION

In a Domain system, every user (and group) is identified by a 64-bit identifier that is unique across all Domain users (and groups) in both space and time. However, UNIX interfaces that take user and group IDs as arguments expect integers that may be unique only within a given file system. The Domain file system stores both forms of identifier with each file. The 64-bit UIDs are used for checking access rights, since there can never be conflicts even if two networks are merged, or a workstation moves from one network to another.

syncids is a program that should be run whenever network registries are merged, or a node is moved between networks having different registries. It ensures that the UNIX owner IDs match the unique owner IDs for every object on the logical volume.

OPTIONS

–a List the name and owner information for each object as it is processed.

–l Only list information about objects for which UNIX owner IDs are incorrect.

–v Verify only; do not actually modify the owners of any objects. This option implies **–l**. If the **–a** option is also given, then information will be printed about every object on the volume.

NAME

telnetd – DARPA TELNET protocol server

SYNOPSIS

/etc/**telnetd**

DESCRIPTION

telnetd is a server which supports the DARPA standard **TELNET** virtual terminal protocol. **telnetd** is invoked by the internet server (see **inetd**), normally for requests to connect to the **TELNET** port as indicated by the /etc/**services** file (see **services**).

telnetd operates by allocating a pseudoterminal device (see **pty**) for a client, then creating a login process which has the slave side of the pseudoterminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the pseudoterminal, implementing the **TELNET** protocol and passing characters between the remote client and the login process.

When a **TELNET** session is started up, **telnetd** sends **TELNET** options to the client side indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal type information* from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudoterminal allocated to the client is configured to operate in ''cooked'' mode, and with XTABS and CRMOD enabled (see **tty**).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **telnetd** is willing to have the remote client *do*: *binary*, *terminal type*, and *suppress go ahead*.

BUGS

Some **TELNET** commands are only partially implemented.

The **TELNET** protocol allows for the exchange of the number of lines and columns on the user's terminal, but **telnetd** doesn't make use of them.

Because of bugs in the original 4.2BSD **telnet**, **telnetd** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2BSD **telnet**.

Binary mode has no common interpretation except between similar operating systems (the UNIX system in this case).

The terminal type name received from the remote client is converted to lowercase.

The *packet* interface to the pseudoterminal (see **pty**) should be used for more intelligent flushing of input and output queues.

telnetd never sends **TELNET** *go ahead* commands.

SEE ALSO

telnet

NAME

tftpd – tftp daemon

SYNOPSIS

/etc/tftpd

DESCRIPTION

tftpd is a daemon which runs the trivial file transfer protocol server for the Aegis Internet software. It is called by **inetd** when an incoming datagram requests the **tftp** service. It then handles **tftp** file transfers in accordance with RFC783.

Note that /etc/**tftpd** must be setuid to a designated **tftp** user (usually a very non-privileged userid and never root) and that the **tftp** spool directory must be owned by that user.

NOTES

The Domain/OS Aegis versions of **tftp** and **tftpd** are adaptations of the Massachusetts Institute of Technology implementations of the **tftp** protocol. Domain/OS Aegis **tftp** will interface with any RFC783 compliant implementation. Note, however, that the 4.3BSD distribution version of **tftp** does not meet these restrictions.

WARNINGS

tftp and **tftpd** comprise an implementation of the Trivial File Transfer Protocol described in RFC783. They allow you to quickly copy files among hosts on an internet without regard to ownership or access restrictions. Therefore, the desired security of a system should be considered before allowing **tftp** transactions. In an inspired attempt to prevent accidental destruction of important files, **tftp** requires that remote file names be absolute pathnames (that is, beginning with /) containing the string ''/**tftp**/'', but not containing the string ''/../''.

SEE ALSO

tftp, inetd;
Configuring and Managing TCP/IP.

NAME

trpt – transliterate protocol trace

SYNOPSIS

trpt [**–a**] [**–c**] [**–d** *<PCB addr>*] [**–e**] [**–f**] [**–j**] [**–l**]
[**–s**] [**–t**] [**–w**] [**–p** *<PCB addr>*] [*<filename>*]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see **setsockopt**), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system, grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

OPTIONS

–a	Print the values of the source and destination addresses for each packet recorded, in addition to the normal output.
–f	Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.
–j	Just give a list of the protocol control block addresses for which there are trace records.
–p *<PCB addr>*	Show only trace records associated with the protocol control block, the address of which follows.
–s	Print a detailed description of the packet sequencing information, in addition to the normal output. (Currently unimplemented)
–t	Print the values for all timers at each point in the trace, in addition to the normal output. (Currently unimplemented)

Domain/OS Aegis EXTENSIONS

–c	Clear trace buffer.
–d *<PCB addr>*	Toggle debug on a connection.
–e	Exit on a bad trace record.
–l	Print lapsed times, in addition to the normal output.
–w	Warn on bad trace records.

NOTES

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the –A option to **netstat**. Then run **trpt** with the **–p** option, supplying the associated protocol control block addresses. The **–f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **–j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a file other than the default, which is **'node_data/systmp/tcp_data,** *<filename>* may be used to specify another file.

DIAGNOSTICS

```
no namelist
```

Printed when the system image doesn't contain the proper symbols to find the trace buffer.

Other diagnostics should be self explanatory.

BUGS

trpt should print the data for each input or output, but this is not saved in the race record.

The output format is inscrutable.

FILES

'node_data/systmp/tcp_data

SEE ALSO

setsockopt, netstat;
Configuring and Managing TCP/IP.

NAME

uctnode – uncatalog a node

SYNOPSIS

/etc/**uctnode** [*options*] *pathname ...*

DESCRIPTION

uctnode removes the specified entry directory name from the local copy of the network root directory. After the name is removed, objects cataloged under that node's entry directory are no longer accessible to you or other nodes on the network.

If you use the **–root** option, the nodename is also removed from the network's replicated root directory.

Node entry directories are created with the **ctnode** command.

pathname (required)	Specify node entry directory name to be uncataloged. Multiple pathnames and wildcarding are permitted.

OPTIONS

–l	List directory names as they are uncataloged.
–root	Uncatalog the node in the network root as well as in the the local root directory.

EXAMPLES

Uncatalog the node with the entry directory name specified.

```
$ /etc/uctnode als_node
```

SEE ALSO

More information is available. Type

help ctnode	For details about cataloging nodes

NAME

uctob –uncatalog the specified pathname, without deleting the associated object.

SYNOPSIS

/etc/uctob [–br] *pathname...*

DESCRIPTION

The command **uctob** removes the specified pathname from the name space. The object associated with the pathname is not affected. This command is primarily intended for system-level debugging use.

OPTIONS

–br	Suppress listing of names and uids of objects as they are uncataloged. These are reported unless this option is specified.

EXAMPLES

```
$ /etc/uctob testfile
"testfile" uid is 16791C0C.40000074.
$
```

This example uncatalogs **testfile**.

SEE ALSO

More information is available. Type

help ctob	For details about cataloging nodes

NAME

ulkob – unlock an object

SYNOPSIS

/etc/**ulkob** [*options*] [*pathname* ...]

DESCRIPTION

ulkob unlocks objects residing on, or locked by processes running on, the current node. You cannot unlock objects on remote nodes unless you locked them (see **–f** below).

This command can be used when a program terminates abnormally, leaving objects locked, or to unlock objects previously locked with the **lkob** command.

To obtain a list of your node's locked objects, use the **llkob** command.

pathname (optional)
Specify name of object to be unlocked. Multiple pathnames and wildcarding are permitted.

Default if omitted: **–u** option must be specified

OPTIONS

If no options are specified, the object is unlocked for all lock modes.

–r	Unlock an object that was locked for read mode; the lock must be owned by this process.
–w	Unlock an object that was locked for write mode; the lock must be owned by this process.
–i	Unlock an object that was locked for reading with intent to write; the lock must be owned by this process.
–f	Forcibly unlock an object. It may have been locked for any mode and the lock may be owned by any process. The object must reside on the current node, however, or must have been locked by the current node. In other words, you cannot unlock objects on a remote node unless you locked them.
–l	List the name of each object as it is unlocked.
–u *uid ...*	Specify the UID of the object(s) to unlock. Multiple UIDs are permitted. If the *pathname* argument is omitted, then this option is required.

EXAMPLES

Forcibly unlock the file **mary** for any mode and unlock the two objects with the specified UIDs.

```
$ /etc/ulkob mary –f

$ /etc/ulkob –uid 1C1A9E2F.20000246 1C1A9E42.50000246
```

SEE ALSO

More information is available. Type

help lkob	For details about locking objects
help llkob	For details about listing those objects that are locked

Appendix A

Using the netmain Interactive Tool and netmain_srvr in the Apollo Token Ring Network

Contents

Appendix A

Using the netmain Interactive Tool and netmain_srvr in the Apollo Token Ring Network

This appendix provides guidelines for using network topology lists and data gathered by **netmain_srvr** and the **netmain** interactive tool displays to draw conclusions about your Apollo Token Ring network's performance. It contains methods to detect conditions that are potentially troublesome, and thus is oriented toward problem prevention. This appendix also contains information on routine maintenance to protect the physical integrity of your network. It includes information about

- Node performance
- Detecting intermittent network events
- Problem isolation methods
- Routine maintenance procedures

A.1 Solving Network Problems with netmain and netmain_srvr

This section provides step-by-step procedures that can help you to start using the **netmain** menus. These procedures show you how to move through the menus and perform some of the basic operations needed to control **netmain_srvr**. Refer to later sections for a full description of the **netmain** interactive tool menus and the **netmain_srvr** options.

A.1.1 Starting and Stopping the netmain_srvr

To start the **netmain_srvr**, use one of the methods described below.

Starting netmain_srvr from the DM Command Line

To start the **netmain_srvr** from the DM command line, type:

Command: /etc/server **netmain_srvr** [*-options*]

The server process begins immediately and continues after logout.

Starting the netmain_srvr from a Start-Up File

To automatically start **netmain_srvr** server, uncomment the following lines in the disked node's **/etc/rc.user** file.

```
#if [ -f  /sys/net/netmain_srvr  ]; then
#  (echo " netmain_srvr\c" >/dev/console)
#  /sys/net/netmain_srvr &
#fi
```

The server process begins when the node comes online and continues after logout. If you don't wish to use the entire set of defaults, you can specify those you wish to use with options in the **netmain_srvr** command line or configuration file. See Section A.4 for details on options.

Stopping the netmain_srvr

When started by the methods described above, **netmain_srvr** continues running until it is intentionally stopped with either of the following commands:

$ **kill netmain_srvr -q** (UNIX operating system environments)

$ **sigp netmain_srvr -q** (Aegis operating system environment)

The server process stops running if the node is shut down intentionally or if the system crashes. If the process stops running, you may start it from the DM command line or by rebooting the node.

A.1.2 Getting Started with netmain_srvr and netmain

Before you can practice using **netmain**, a monitor has to run in your network. If no monitors are running in your network, start **netmain_srvr** as described above. Let the monitor run, with default values, for a week. Then you may begin Procedure A-1. If you already have had **netmain_srvr** running for more than a week, proceed directly to Procedure A-1.

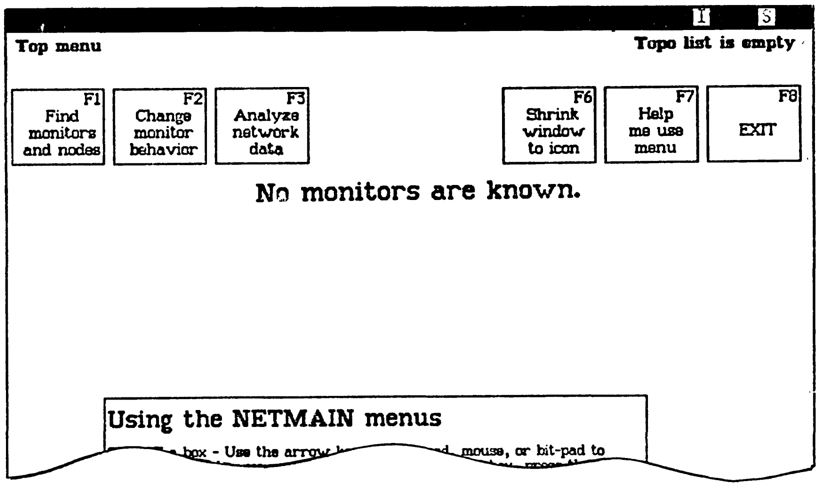
Procedure A–1 *Getting Started with Netmain*

Task 1: Invoke netmain.

To invoke **netmain**, type:

```
$ netmain –whelp
```

The **–whelp** option tells **netmain** to grow the window, to display the entire **netmain** menu, including the help area. The Top–Level menu, the top of which is shown below, appears.



No monitors are known.

Using the NETMAIN menus

box - Use the arrow mouse, or bit-pad to

Figure A–1. Top–Level Menu

Task 2: Study the menu.

The **menu name** appears in the top–left corner.

Each **command box**, across the top of the menu, shows a command and the keyboard **function keys**, F1 through F8.

Netmain messages that describe functions selected appear below the command boxes. The message, “No monitors are known,” below the command boxes reminds you that you haven’t yet chosen a monitor. **Netmain** also uses this area to draw an input box and to prompt you for input.

Error and status messages appear above the command boxes. **Netmain** reports on its activity as it carries out commands. There are no messages above the command box right now. A “Topology List status” message appears in the top right corner.

The large box at the bottom of the screen contains online help about each menu.

Task 3: **Locate the + shaped cursor in the netmain window.**

Use the arrow keys, touch–pad, mouse, or bitpad to move the cursor. Move the cursor outside the window. It regains its rectangular shape and the Display Manager is in control. Move the cursor back to the **netmain** window.

Taks 4: **Point to a command box by placing the cursor in the box.**

Notice that pointing to a command box accents the box. It changes color. When a box is accented, the + shaped cursor is no longer visible.

Task 5: **Point to the "F6 Shrink window to icon" command box to make the netmain window into an icon.**

To get help on any **netmain** command or parameter, type **h**, or **?**, or press the middle or right buttons on the mouse, or press any button except the top on the bit–pad puck. The menu help box displays additional information about the command.

Task 6: **Select the "F6 Shrink window to icon" command.**

To select a box in a **netmain** menu, you can use one of the following methods:

- Press the appropriate function key (only for command boxes labeled with function key names).
- Point to the box and press the keyboard's space bar.
- Point to the box and press the mouse's left button.
- Point to the box and press the top button of the bit–pad puck.

The **netmain** window becomes an icon. The **netmain** program continues to run in the icon, retaining any information it has amassed, but taking up little space on the screen.

Task 7: **Point to the netmain icon by moving the DM cursor over the icon.**

Task 8: **Request help about the icon by pointing to it and typing h or ?, pressing the middle or right button on the mouse, or pressing any button except the top one on the bit–pad puck.**

A system help file is displayed.

Task 9: **Enter netmain by pointing to the icon and pressing the space bar. Then select "F2 Change monitor behavior."**

Note the error message that appears. You have not yet selected a monitor with which to work.

Task 10: **Now look for the command box to exit from netmain.**

As you probably guessed, it's the F8 Exit box. On any **netmain** menu, use this command box to exit to the higher–level menu. On this top menu, use it to exit from **netmain**.

A.1.3 Establishing Network Performance Levels

If users perceive poor network performance, use the information **netmain_srvr** gathers to determine the reason. Sometimes a small or intermittent problem in a connector, cable, or ring interface board can cause network or node performance problems. This section describes how to use the **netmain** interactive tool to look for such problems and isolate them to one point in the network, usually a node's IN and OUT cables and connectors. Sometimes poor distribution of network resources can also cause poor network performance. This section describes how to "tune" the network by searching for subtle problems that can affect performance. Follow the suggestions in this section to prevent problems and enhance network performance. Section A.5 describes how to use the **netmain** interactive tool to detect resource distribution problems.

To increase your understanding of the material in this appendix, use the **netmain** interactive tool and perform the operations described as you read the material. Allow ample time to cover the material, and keep your reading and practice sessions short.

Evaluating Node Performance

The **netmain** interactive tool displays can be used to analyze individual node performance as well as overall network performance. This section describes how to use these displays to evaluate the performance of individual nodes in the network. Questions that require an evaluation of node performance include the following:

- What percentage of network resources are provided by individual nodes, and is this distribution appropriate for the network?
- Are nodes' diskless partner assignments affecting the performance of either node?
- Is any node experiencing a high number of disk or memory errors that could indicate hardware problems?

Locating Underused or Overused Nodes

The CPU_TIME and NET_SERVICE probes gather data about CPU usage and network requests. The categories CPU SERVICE and QUEUE SERVICE from the **netmain** interactive tool's Analyze Network Data menu include performance statistics for data gathered by these probes. To locate underused or overused nodes, look at performance statistics in these categories. Also, use the output formats that report on diskless nodes. This section provides guidelines for using these tools.

Look at the network resources that nodes provide by checking the number of diskless partners assigned to each node. Choose the "Diskless partners" output format and select "Partners, by mother." Check the number of diskless partners assigned to the nodes shown. If some nodes have several diskless partners and others have none, consider reassigning some diskless nodes to other partners. If the nodes are servers, dedicated to serving diskless node, the arrangement may be satisfactory.

Next, choose a "Density across network" output format. It shows each node's relative contribution of a resource as a percentage of the total for the network. Start by looking at the "Total file service" and "Total page service" performance statistics. These statistics show the number of file services and paging services each node performs for other nodes. A node provides these services when other nodes need to read, write, open, close, and list its files. Nodes with diskless partners have higher counts of these services than nodes without diskless partners. Experiment by setting the "top of error scale" level in the Parameter menu to different values. In the output displays, look for the following patterns:

- Sharp, high-density, vertical lines underneath certain nodes, extending across all time intervals
- Sharp, high-density, vertical lines that appear only during certain time intervals

A node with a high-density vertical line is heavily consuming the network resource shown by that performance statistic. Check to see whether such heavily used nodes are servers nodes (typically DSPs, such as the DSP90 or DSP160), and/or whether they have diskless partners (use the "Diskless partners" format).

If a heavily used node has no diskless partners and isn't a server node, it may contain a system resource that can be replicated elsewhere. For example, if commonly used sets of files can be replicated on other nodes, the traffic at this node will be lighter. Note, however, that files are good candidates for replication only if there is a mechanism for coordinating file updates. If a node is heavily used only at certain times (a high-density line appears at certain time intervals, then fades), investigate the activity on that node during these times.

If only servers and/or partners of diskless nodes are used heavily, check the relative contribution of each server. Create a data file containing only the names of the servers. Use the format described in Section A.5 for the F5 (Topo list from data file) command in the Find Monitors and Nodes menu. Put the data file in the topology list. Then use the "Density across network" format to look at the performance statistics again.

Look for high–density vertical lines that indicate heavy loads on individual nodes and also look for high–density horizontal bands. High–density bands can give information about times of peak use in the network. Analyze user activities during these peak periods to determine which activities consume system resources. You may be able to change the "mix" of services on the server nodes to create a more efficient pattern of use.

Use rates or incidence formats to analyze nodes' file and paging services performed for other nodes. To find out about a node's performance during times of peak use, display "page or file backlog severity" performance statistics (from the QUEUE_SERVICE category). Backlog severity statistics report the average number of service requests in the file or paging service queues, at times when there are requests in these queues.

These statistics show the nodes that are most heavily used during peak periods. Traffic in local area networks tends to occur in bursts, and distributing resources to share the load during peak periods can help to provide maximum performance of your network. To compare the contributions of individual nodes, use an across–network display format.

The "any backlog" statistic shows all unserviced requests in a service queue. The "average backlog" statistic shows the average number of requests in a queue over time. Note that a high average backlog or backlog severity level is not an error condition. It shows that the node is providing a relatively high percentage of service to the network. Evaluating this condition depends on what you intend the node to provide, and what you intend other nodes to provide.

If an "any backlog" display shows that some nodes have backlogs and others do not, distribute more resources to nodes without backlogs. If some nodes have a higher average queue length than others, investigate the number of pages of memory that node has allotted for paging requests from remote nodes. If a user has used the **netsvc** shell command with the **–p**[*n*] option, remote nodes are limited to only *n* pages of the node's memory, and this could add slightly to the queue length.

For more information when you suspect a node is overused, look at the "Total disk activity" performance statistic from the **DISK** category. Display the data by using the same rates or incidence formats that you used with the CPU SERVICE and QUEUE SERVICE data. Use the Display Manager (DM) to place one output pad directly over the other. If both displays show a vertical high–density line for the node in question, it is probably overused. To confirm this, compare this node's **DISK** category statistic with those of other nodes in the network. If this node's disk activity is comparable to that of other nodes, it is not, in fact, being overused.

The procedures described above are used to search for nodes that are providing more than their share of network resources. To find nodes that are providing less than their share, use the "Null CPU time" performance statistic in various display formats. Density formats are useful because they allow you to easily compare node's performance.

All nodes have some percentage of Null CPU time. Compare the node's "Null CPU" time to its "Total disk activity" from the **DISK** category, using the "two–display" technique described earlier. If the node shows both a high disk activity level and a high Null CPU time level, the node may be spending too much time servicing paging requests and not enough time using the CPU for computation. The node's performance might improve if it were used less as a system resource.

Nodes that show high "Null CPU" time but do not have a high "Total disk activity" level can provide more network services. Nodes without service queue backlogs can also provide more network services. Use a "peaks" format to look for nodes with no service queue backlogs.

Diskless Partner Information

The previous subsection described a quick check to compare diskless partner assignments for nodes of various types. Use the network resource analysis techniques described in this subsection to ensure that nodes are not acting as partners to too many diskless nodes.

To ensure that diskless nodes' partners are in the same network loop, prepare data files with lists of nodes in each network loop. Put the data files into the Topo List (F5 in the Find Monitors and Nodes menu). Then use the "Diskless partners" output format to display diskless nodes and their partner nodes. Verify that each diskless node has a partner in the same loop.

Disk and Memory Errors

Performance statistics in the **DISK** and **STORAGE MODULE** categories, and printed displays on memory errors can alert you to potential hardware problems. In the **DISK** and **STORAGE MODULE** categories, use any graph display format, particularly a peaks format, to check for Disk or SMD (Storage Module Disk) CRC error. Set the "top of error scale" to 5%. If a disk or SMD shows high or increasing levels of CRC errors (above 0.01%), copy user files elsewhere and contact a service representative. Incidences of "Disk/SMD not ready" or "Disk equipment check" conditions also should not occur.

In printed displays of memory errors ,look for sudden occurrences of ECCC or ECCU errors. If these occur, contact a service representative. In the case of ECCC errors, the service representative can determine if there actually is a problem. In the case of ECCU errors, a hardware problem is usually indicated. The service representative may replace a failing board.

A.2 Detecting Unusual or Intermittent Network Events

Use the methods described below to find intermittent or unusual network performance conditions and to recognize these conditions in **netmain** tool displays.

Become familiar with the performance levels typical of your network. Do this by producing plots via the "incidence density output" format. Choose performance statistics in the **RING RECEIVE** or **RING TRANSMIT** categories. Look for horizontal high–density bands extending across all nodes in the network for a period of time. These high–density bands indicate that the entire network showed increased levels of the performance statistic you are analyzing, over a certain time period.

Compare the times at which horizontal high–density bands appear on plots for different days. If there is any pattern to the time periods, investigate user and network activities occurring during these time periods.

When you start to look for intermittent conditions, check two performance statistics in the RING TRANSMIT category, "Transmit no return" and "Transmit packet error," for levels that rise across the entire network during any kind of a problem.

Select "Transmit no return" and an incidence density format to display the data. Set the "top of error scale" to 25%. If most of the nodes in the network show some gray, there may be a problem worth investigating further. Increased levels of this statistic occur when nodes send a packet but do not get a response in return. High levels for this statistic can indicate a cable or connector problem. Cable or connector problems can cause nodes to send hardware failure messages. Select a "Scattergram events output" format and look at hardware failure messages. Note the nodes that report hardware failure messages. Check the cables and connectors for these nodes and for nodes directly upstream of them.

If there are no hardware failure messages in the scattergram, or if the "Transmit no return" statistics did not indicate problems, look at the "Transmit packet error" performance statistic. Choose an "incidence density" format to display the data. Leave the "top of error percentage scale" at its default value of 50%. If most of the nodes in the network don't show much gray, there is probably not a problem. If most do show gray, choose an "incidence peaks" format and experiment with various "Top of error threshold scale" levels. Note the level at which many nodes start to show peaks of "Transmit packet errors." Also note whether the peaks occur for most nodes or only some nodes.

Isolating a Problem to a Particular Node

This subsection describes how to look at displays of performance statistics to isolate the source of the problems described in the previous subsections.

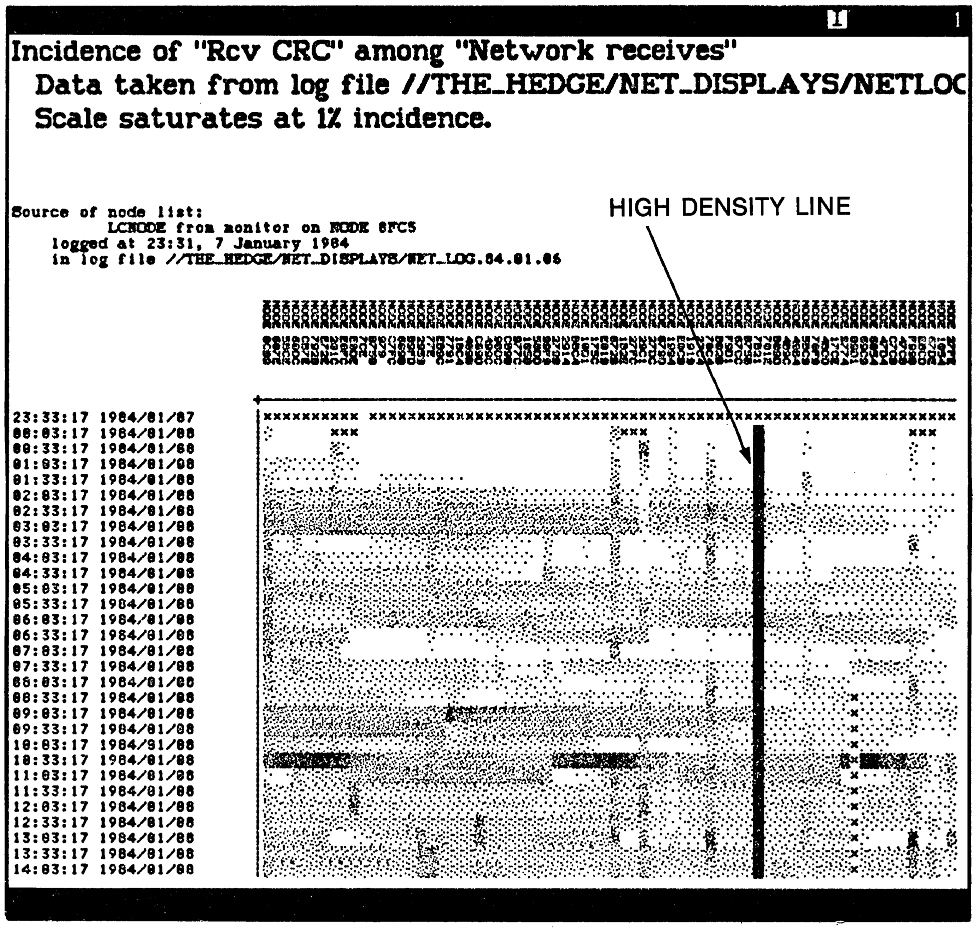
Choose an "incidence density" format and any of the following performance statistics:

- Receive modem errors (from RING RECEIVE category)
- Receive biphase errors (from RING RECEIVE category)
- Transmit modem errors (from RING TRANSMIT category)
- Transmit biphase errors (from RING TRANSMIT category)

In the displays, look for a vertical, sharp, high–density line under one node. It indicates saturation or near–saturation conditions for that node. If you don't see a high–density line under any node in the gray scale plot, lower the "top of error scale" and see if the condition occurs. Figure A–2 shows an example of such a condition in a density plot.

If the output shows a high–density vertical line, inspect that node, the node immediately upstream, and all the cables and connections between these nodes. This output can indicate a problem in the node's ring interface hardware that is not affecting other nodes in the network, but sometimes it can point out a problem with cabling or a connection.

Check further by examining output from an active monitor while someone moves the cables in the areas that you suspect. If the count increases, the problem is probably in the physical medium (cables or connectors).



```
Incidence of "Rcv CRC" among "Network receives"
  Data taken from log file //THE_HEDGE/NET_DISPLAYS/NETLOC
  Scale saturates at 1% incidence.

Source of node list:
            LCNODE from monitor on NODE 8FC5
    logged at 23:31, 7 January 1984
    in log file //THE_HEDGE/NET_DISPLAYS/NET_LOG.04.01.06

23:33:17 1984/01/07
00:03:17 1984/01/08
00:33:17 1984/01/08
01:03:17 1984/01/08
01:33:17 1984/01/08
02:03:17 1984/01/08
02:33:17 1984/01/08
03:03:17 1984/01/08
03:33:17 1984/01/08
04:03:17 1984/01/08
04:33:17 1984/01/08
05:03:17 1984/01/08
05:33:17 1984/01/08
06:03:17 1984/01/08
06:33:17 1984/01/08
07:03:17 1984/01/08
07:33:17 1984/01/08
08:03:17 1984/01/08
08:33:17 1984/01/08
09:03:17 1984/01/08
09:33:17 1984/01/08
10:03:17 1984/01/08
10:33:17 1984/01/08
11:03:17 1984/01/08
11:33:17 1984/01/08
12:03:17 1984/01/08
12:33:17 1984/01/08
13:03:17 1984/01/08
13:33:17 1984/01/08
14:03:17 1984/01/08
```

Figure A–2. High–Density Line Condition

If moving the cables does not affect the density, the problem may be in the ring interface hardware for the node that showed the high–density vertical line, or the ring hardware of the node upstream. Use the shell command **netsvc –n** to take one of the nodes off the network while you continue to examine output from an active monitor. If the high–density vertical line starts to disappear, the problem is probably in the node that you removed from the network. While the node is off the network, see if "Transmit packet error" levels go down significantly. If the results show that the node is hindering network performance, schedule maintenance for the node.

Choose an "incidence density" format, and the "Rcv CRC" performance statistic in the RING RECEIVE category. The operating system increments counts for this statistic when part of a received message does not pass the CRC. Thus, this statistic correlates with corruption of one or several bits in a packet.

Set the "Top of error percentage scale" to 1%. Look for areas of high density that fade horizontally into areas of lower density. Figure A–3 shows an example of high–density fade in a gray scale plot.

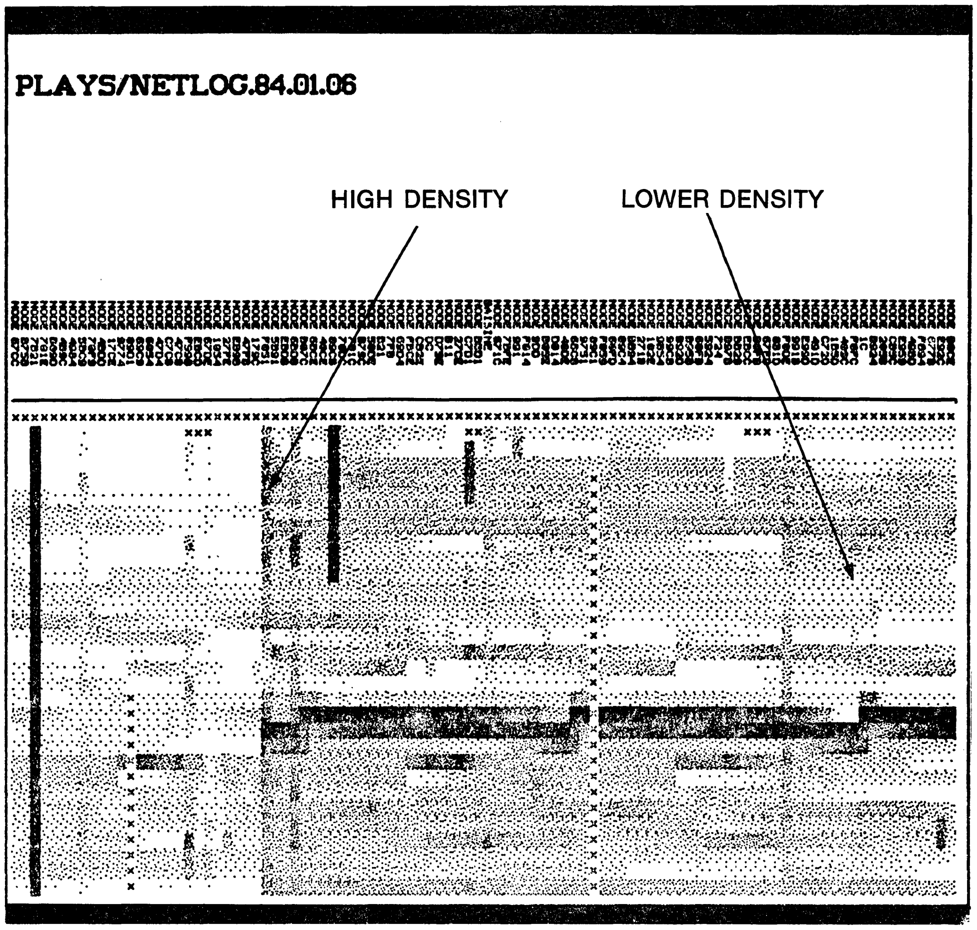


Figure A–3. High–Density Fade Condition

High–density areas that fade into low–density areas indicate a problem in one node that affects data integrity in other nodes. The nodes with the highest density are closest to the problem because almost all the messages they receive must pass through the node causing the problem. Nodes that are further downstream show a lower error density because they receive some messages that haven't passed through the node causing a problem.

If you do not detect the condition, set the "Top of error scale" lower. The high–to–low density fade condition can be seen more readily in larger networks since many more nodes are sampled. If you detect the condition, put the output pad for the plot near the bottom of the screen. Then request another incidence density display for the "Transmit modem error" performance statistic. Set the plot resolution to the same level you used for the "Rcv CRC error" performance statistic. Move the second output pad so that its columns line up with the first. Look at the node at which the high–density fade begins in the "Rcv crc" plot. Check for a high-density vertical line located under the same node in the "modem error" plot. In this situation, the node is producing modem errors, which corrupts data in nodes downstream. However, the further away a node is from the problem node, the fewer of its received messages must pass through the problem node, so the lower its percentage of corrupted data. The increasingly lower percentages produce the fade effect. Figure A-4 shows two plots aligned in just this manner.

If you isolate a problem node by aligning plots, investigate the node, the node upstream, and their cables and connectors very thoroughly. Use this technique with "Density across network" plots as well as incidence density plots. In addition, look for density fade conditions for "Rcv ack parity" and "Rcv header checksum" performance statistics.

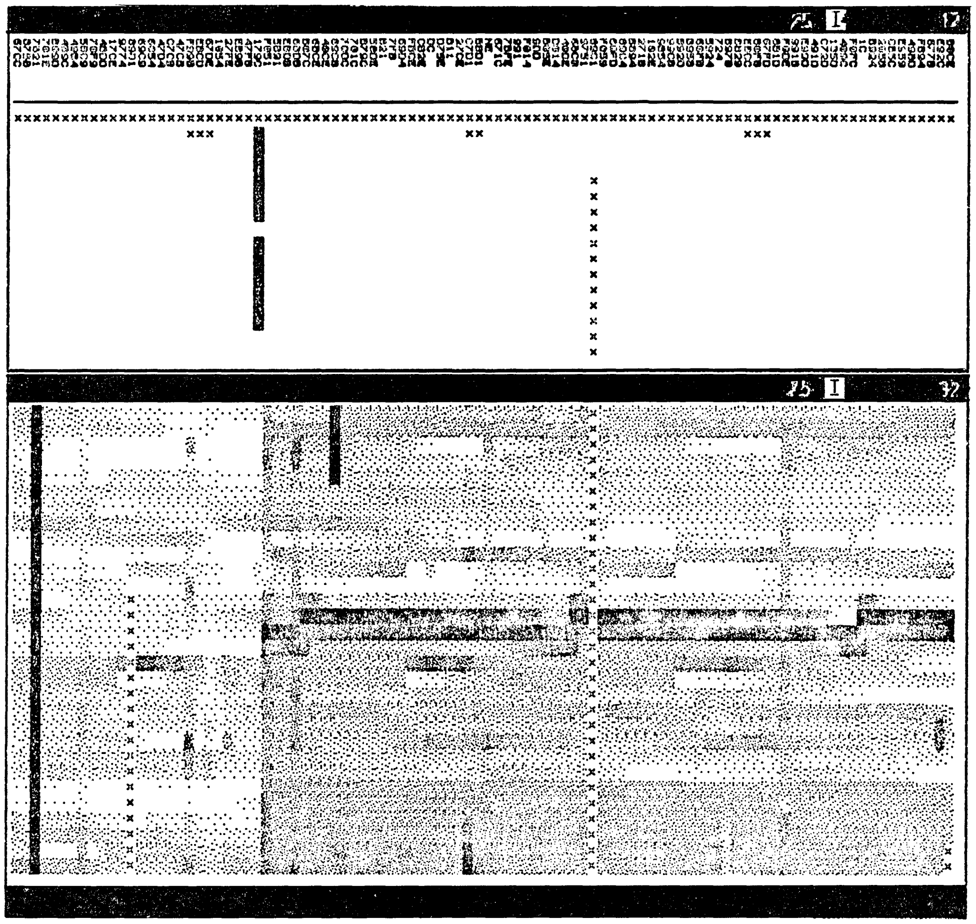


Figure A-4. Aligned Plots

More Intensive Methods of Locating Network Performance Problems

If you want more information about a node than is provided by the methods described previously, activate the SWD_10_MSGS or SWD_100_MSGS probe on a monitor in the network. These probes record performance statistic levels before and after transmission of a set of messages to each node. The SW DIAGNOSTIC (SWD) category includes many of the performance statistics that are in the RING RECEIVE category, but the counts displayed are only about packets sent by SWD messages.

To analyze SWD performance statistics, use the techniques described in the preceding sections. Be aware, however, that these probes can significantly add to network traffic. Use them to get information unobtainable by other means.

A.3 The Network Log Book

Keep a network log book, containing the information collected from **netmain_srvr**, online or as a hard copy. In it, store files produced from the **netmain** interactive tool displays and other relevant network information. Record the date and time that new nodes are installed in the network, and identify the nodes that run network services.

Include in the log book the following information about network problems:

- Date and time the problem was detected (essential in tracking network problems and helpful in interpreting the information gathered by **netmain_srvr**)
- Name of person who discovered and/or debugged the network problem
- The node that reported a ring hardware error, if any, or from the **netmain** interactive tool's Analyze Network Data menu
- The **netmain** interactive tool analysis information that relates to the problem
- Action taken to restore the network, if necessary

Write supplemental notes to **netmain_srvr** log files. Include information about scheduled service time, new node installations, and problem occurrences. Write notes in two ways:

- With the **netmain** interactive tool, use the "F4 Log a text string" command box in the Change Monitor Behavior menu.
- Outside the **netmain** interactive environment, use the **netmain_note** command, as shown below.

Type:

```
$ /com/netmain_note message
```

For example, you might type:

```
$ /com/netmain_note Loop 17 scheduled downtime
```

To retrieve the notes stored in a log file, use the "Printed Output" format in the **netmain** interactive tool's Analyze Network Data menu.

A.4 The netmain_srvr Reference

The **netmain_srvr** (**/sys/net/netmain_srvr**) program collects data used for maintaining the Apollo Token Ring network. Use the data for monitoring ring performance and isolating potential problems. Use the **netmain** interactive tool, described in Section A.5, to interact with **netmain_srvr**.

The **netmain_srvr** process running on any node is called the node's monitor. Monitors execute subprograms called "probes" and "observers", which are usually in a waiting state. At specified intervals, each probe becomes active and collects some data about nodes in the ring. The probe stores the information it collects in non-ASCII files called "log" files, then returns to its waiting state.

Each time a probe gathers data, an observer sifts through it to detect either a transmit modem error (**MODEM_ERRS**) or a disk drive error (**WIN_CRC**). If an observer detects either condition, it can provide an alarm on the node running a monitor.

The **netmain_srvr**'s probe and observer log files on active monitors are open. You cannot analyze the information in them until you close them with the **netmain** interactive tool. Log files reside in the **'node_data/system_logs/net_log** directory by default.

If netmain_srvr does not start properly, a record of the failure appears in the file **'node_data/system_logs/netmain_srvr.err.log**. If you experience problems starting **netmain_srvr** or if the monitor dies, use the data in the error log to determine the cause of the problem. The **netmain_srvr** error logs are ASCII files, and you can treat them as you would treat any ASCII file.

There are two shell commands that can help you manage **netmain_srvr** logs. The first, **netmain_chklog**, checks for and, if appropriate, deletes corrupted **netmain_srvr** log files. Use **netmain_chklog** if a node running **netmain_srvr** crashes or is reset (via the RESET switch or button). The log file that **netmain_srvr** was writing when the node crashed will be corrupted. Delete it with **netmain_chklog**. Attempts to analyze data from a corrupted log file may cause **netmain** to behave unpredictably.

Use the **netmain_note** shell command when you are outside of the **netmain** interactive tool environment and want to send a text string to a **netmain_srvr** log file. The *Command Reference* manual for any of the three environments provides information on **netmain_chklog** and **netmain_note**.

A.4.1 Invoking netmain_srvr

The **netmain_srvr** process should run as a background process from the **etc/rc** file. By default, the process names itself **netmain_srvr**, so you need not use the **–n** option with the process creation command. Run monitors only on nodes with disks, not on diskless nodes or Domain Server Processors (DSP)s. Whenever possible, run a monitor in each loop so that data can be collected even if the loop is switched out of the main ring. Ensure that monitors are configured so that they collect the data you need, without creating log files that take up too much disk space, and without affecting performance of the nodes on which the monitors run.

Online help for instructions on controlling **netmain_srvr** after it starts, and on analyzing the data collected is available by typing

```
$ /com/help netmain
```

For information about adding notes to the network error log, type

```
$ /com/help netmain_note
```

For information about detecting and deleting corrupt log files, type

```
$ /com/help netmain_chklog
```

To start **netmain_srvr** from the DM command line, type

```
Command: /etc/server netmain_srvr  [–options]
```

The process begins immediately and continues after logout.

To automatically start the server, uncomment the following four lines in the disked node's **etc/rc.user** file.

```
#if [ -f  /sys/net/netmain_srvr  ]; then
#  (echo " netmain_srvr\c" >/dev/console)
#  /sys/net/netmain_srvr &
#fi
```

The process begins when the node comes online and continues after logout. If you don't wish to use the entire set of defaults for a monitor, you can specify those you wish to use with options in the **netmain _srvr** command line or configuration file.

When started by the methods described above, **netmain_srvr** continues running until it is intentionally stopped with either of the following shell commands:

$ kill netmain_srvr -q (UNIX operating system environments)

$ sigp netmain_srvr -q (Aegis operating system environment)

The process stops running if the node is shut down intentionally or if the system crashes. If the process stops running, you may start it from the DM command line or by rebooting the node.

Options and Arguments

The **netmain_srvr** process has a number of options. Instead of including them all on the command line you can use an options file by specifying the **-c[mdf]** option. If you specify **-c** pathname, the server first reads the options listed in the options file specified, and then reads any other options on the **netmain_srvr** command line. If there are any conflicts between the options file and the command line, the command line settings are used. For example, if the options file specifies **-ll 1500** and the command line specifies **-ll 3000,** 3000 is the limit on the log file's length.

Default options are indicated by (D).

-a[ppend] Appends to an existing log file the name specified by the **-l** option; otherwise, creates a log file with this name. This option is only valid when a log file pathname is specified with the **-l** option. Contrast this with the **-nappend** option.

-c[mdf] [*pathname*]
Accepts options from an ASCII text file pathname. You may use this option only from the command line, not in the options file. There can only be one options file.

-l[og] [*pathname*] (D)
Creates a log file. Optionally, this specifies a pathname, which is rela-

tive to the **‘node_data/system_logs/net_log** directory. If either this option or the pathname is not specified, the log filename is derived from the current date: **‘node_data/system_logs/net_log/net_ log.yy.mm.dd**. The log file is stored on the disk of the node running **netmain_srvr** and must remain there for **netmain_srvr** to write to it.

A.4.2 Data Collected by netmain_srvr Probes and Observers

The following list describes the data collected by the probes and observers run by **net-main_srvr** monitors. The descriptions of items sometimes refer to "output formats," or "display formats." Generate these formats with the **netmain** interactive tool.

CPU_TIME – Null/AEGIS/user CPU Time

The CPU_TIME probe records performance statistics about each node's CPU usage and writes them to **‘node_data/system_logs/net_log/net_log.yy.mm.dd** or a file you specify.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Aegis CPU time:	Shows the time the operating system on each node spends on internal needs, and on servicing requests from other nodes. Expressed as a percentage of elapsed time. This statistic does not show the time the operating system spends servicing calls from user programs, running server programs, or running the Display Manager.
Null CPU time:	Shows the time each node's CPU is idle, as a percentage of elapsed time. Idle CPU time is any time during which the CPU is not performing computations. Expect all nodes to show a certain amount of idle time, for time spent servicing page faults, etc. If a node shows both a high disk activity level and a high CPU time level, it may be spending too much time servicing page faults (thrashing).
User CPU time:	Shows, as a percentage of elapsed time, the time the CPU on each node spends running user programs, shell programs, the Display Manager, and server processes. The statistic does not currently show the time used by individual processes or programs.

DISK_ERRS – Disk and Storage Module Errors

The DISK_ERROR probe records cumulative information about disk and storage module performance and errors on all nodes and writes them to **‘node_data/system_logs/net_log/ net_log.yy.mm.dd** or a file you specify.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Controller busy:	Counts the number of requests for disk I/O that could not be serviced because the controller was busy. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.
CRC errors:	Counts Cyclic Redundancy Check (CRC) errors on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.
Disk reads as percentage of disk I/O:	Calculates the ratio of reads to the node's total Winchester disk I/O. The performance statistic is shown as a percentage of the node's total Winchester I/O. If the count for this statistic is 49 percent, for example, reads account for 49 percent of all the node's Winchester I/O.
Disk writes as percentage of disk I/O:	Calculates the ratio of writes to the node's total Winchester disk I/O. The performance statistic is shown as a percentage of the node's total Winchester I/O. If the count for this statistic is 51 percent, for example, writes account for 51 percent of all the node's Winchester I/O.
Equipment check:	Counts the number of device equipment checks that occur. Problems on the device controller, such as controller memory component errors, internal micro-diagnostic failures, or internal timing problems, can cause equipment checks. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.
Not ready:	Counts the number of times that a "disk not ready" error condition occurs on a disk. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.
Overruns:	Counts Direct Memory Access (DMA) overruns on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.
Seek error:	Counts the number of attempts to find a track that occur on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.
Storage module reads as percentage of storage module I/O:	Shows the ratio of storage module reads to the node's total storage module I/O. The performance statistic is shown as a percentage of the node's total storage module I/O. If the count for this statistic is 51 percent, for example, reads account for 51 percent of all the storage

module's I/O. Only data for the first storage module (Unit 0) is displayed. Counts are expressed as a percentage of the device's I/O.

Storage module writes as percentage of storage module I/O:
Shows the ratio of storage module writes to the node's total storage module I/O. The performance statistic is shown as a percentage of the node's total storage module I/O. If the count for this statistic is 51 percent, for example, writes account for 51 percent of all the storage module's I/O. Only data for the first storage module (Unit 0) is displayed.

Timeouts:
Counts the number of controller timeouts on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Total storage module activity:
Measures the total disk storage module activity (reads plus writes) of each node (always 0 for nodes without storage modules). Use this performance statistic only with plots of network totals or rates from the **netmain** interactive tool. Shows the disk activity per node as a percentage of the total Winchester disk activity for the entire network, or absolute I/O rates per unit time. Only data for the first storage module (unit 0) is displayed.

Total Winchester disk activity:
Measures the total Winchester disk activity (reads plus writes) of each node. Use this performance statistic only with plots of network totals or rates from the **netmain** interactive tool. Shows the disk activity per node as a percentage of the total Winchester disk activity for the entire network, or absolute I/O rates per unit time.

ERR_COUNTS – Network Error Counts (Normal Traffic)

The ERROR_COUNTS probe records cumulative network performance statistics and error counts on all nodes and writes them to **'node_data/system_logs/net_log/net_log.yy.mm.dd** or a file you specify.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Acknowledge, negative (NACK):
Counts the number of transmissions in which the destination node did not acknowledge receipt of the message. NACKs can occur when nodes send messages to a node whose loop is switched out of the network, or a node that is not running the operating system. Expect large numbers of NACKs on nodes running **netmain_srvr**. The display

formats show the statistic as a percentage of the node's total network transmits.

Acknowledge parity, receive (ACK):
Counts the number of parity errors in the hardware protocol segments of received messages. The hardware always detects ACK parity errors if any occur. The display formats show the statistic as a percentage of the node's total network receives.

Acknowledge parity, transmit (ACK):
Counts the number of parity errors in the hardware protocol segments of transmitted messages. The hardware always detects ACK parity errors if any occur. The display formats show the statistic as a percentage of the node's total network transmits.

Acknowledge, Wait (WACK):
Counts the number of times that the destination node was too busy to accept a message in the time allotted. This performance statistic does not necessarily reflect an error condition — on a DN4xx/DN6xx nodes, for example, the node may have been performing disk I/O using the shared ring/disk hardware. The display formats show the statistic as a percentage of the node's total network transmits.

Biphase error, receive:
Counts the number of biphase errors that occur. Biphase errors occur when a node's modem hardware cannot lock on the transmission frequency from the node upstream. Biphase errors can result from modem hardware failures, broken cables or connectors, or signal degradation caused by excessive cable lengths between active nodes. The display formats show the statistic as a percentage of the node's total network receives. The ERR_COUNTS probe collects data for this statistic only from nodes running SR8 or later releases.

Biphase error, transmit:
Counts the number of biphase errors that occur. Biphase errors occur when a node's modem hardware cannot lock on the transmission frequency from the node upstream. Biphase errors can result from modem hardware failures, broken cables or connectors, or signal degradation caused by excessive cable lengths between active nodes. The display formats show the statistic as a percentage of the node's total network transmits. The ERR_COUNTS probe collects data for this performance statistic, only from nodes running SR8 or later releases.

Bus error, receive: Counts errors during Direct Memory Access (DMA) from the ring controller. The display formats show the statistic as a percentage of the node's total network receives.

Bus error, transmit:	Counts the number of times that the ring controller experienced a bus error during a Direct Memory Access (DMA) transfer. The display formats show the statistic as a percentage of the node's total network transmits.
CRC, receive:	Counts messages that contain Cyclical Redundancy Check (CRC) errors, detected by the ring hardware. CRC errors can result from errors in any part of the received message, except the hardware protocol segments. You cannot disable CRC checking. Compare CRC error statistics to those for RCV header checksum and ACK parity errors. The display formats show the statistic as a percentage of the node's total network transmits.
End–of–range, receive:	Counts the number of times that one or both of the message fields in the packet received was larger than the Direct Memory Access (DMA) channel allowed for it. The display formats show the statistic as a percentage of the node's total network receives.
ESB errors, receive:	Counts the number of elastic store buffer errors received. These errors arise when the node cannot follow a large or sudden change in the network communication frequency. The display formats show the statistic as a percentage of the node's total network receives. The ERR_COUNTS probe collects data for this statistic only from nodes running SR8 or later releases.
ESB errors, transmit:	Counts the number of Elastic Store Buffer (ESB) errors that occur. ESB errors occur when the node is unable to follow a large or sudden change in the network's communication frequency. The display formats show the statistic as a percentage of the node's total network transmits. The ERR_COUNTS probe collects data for this statistic, only from nodes running SR8 or later releases.
Header checksum, receive:	Counts messages that contain checksum errors in the message header. Since the operating system program that verifies header checksums is usually disabled, the count for this statistic should be 0. The display formats show the statistic as a percentage of the node's total network receives.
Modem error, receive:	Counts the number of times the receiver could not synchronize properly with the network. The conditions that cause this error are biphase or Elastic Store Buffer (ESB) errors. See the "receive biphase error" or "receive ESB error" statistics. The display formats show the statistic as a percentage of the node's total network receives.

Modem error, transmit:	Counts the number of times the transmitter could not synchronize properly with the network, resulting in an Xmit ESB or biphase error condition. See "transmit biphase errors" and "transmit ESB errors." If the error condition lasts more than one minute, the node broadcasts a "hardware failure report" (displayed in the nodestat shell command output). A broken cable can cause modem errors. The display formats show the statistic as a percentage of the node's total network transmits.
No return, transmit:	Counts the number of transmitted messages that failed to return to the node. After a destination node receives and copies a message, the message continues to travel around the ring until it returns to its transmitter, which removes the message. If a message does not return, it could not complete the loop around the ring, indicating a break in the ring. The display formats show the statistic as a percentage of the node's total network transmits.
Overrun, receive:	Counts the number of Direct Memory Access (DMA) overruns that occur. The display formats show the statistic as a percentage of the node's total network receives.
Overrun, transmit:	Counts the number of times that the ring controller experienced a Direct Memory Access (DMA) overrun condition during a ring transmit. The display formats show the statistic as a percentage of the node's total network transmits.
Packet error, receive:	Counts the number of times either the receiver or the transmitter had problems with messages. If the fault was in the receiver, other counts are incremented also. The display formats show the statistic as a percentage of the node's total network receives.
Packet error, transmit:	Counts the number of times an error occurs during transmission of a message. The system increments counts for this statistic when other error conditions occur. The display formats show the statistic as a percentage of the node's total network transmits.
Timeout, receive:	Counts messages received that did not complete in the expected time. The display formats show the statistic as a percentage of the node's total network receives.
Timeout, transmit:	Counts transmitted messages that do not complete their transmission in the expected time. This error often occurs when network traffic is slow, due to repeated attempts to retransmit or regenerate the ring token. The display formats show the statistic as a percentage of the node's total network transmits.

Transmit call: Counts the number of requests to transfer data out of the node, on to the ring. The transmit call counter is increased even if the actual transmit fails. This performance statistic does not reflect any error conditions. If the number of requests is less than 100 percent of the ring transfers attempted, the node had to retry some of the transfers. The display formats show the statistic as a percentage of the node's total network transmits.

Transmit error, receive:
Counts the number of times that either the transmitter or another receiver had an error in the packet. For this error to occur, some other error flag must be set. The display formats show the statistic as a percentage of the node's total network receives.

Receives as percentage of network I/O:
Calculates the ratio of incoming messages (receives) to the node's total network I/O. The performance statistic is shown as a percentage of the node's total network traffic. If the count for this statistic is 43 percent, for example, incoming messages account for 43 percent of all the node's network I/O activities.

Sends as percentage of network I/O:
Calculates the ratio of transmitted messages (sends) to the node's total network I/O. The performance statistic is shown as a percentage of the node's total network traffic. If the count for this statistic is 43 percent, for example, outgoing messages account for 43 percent of all the node's network I/O activities.

Total network activity:
Measures the total network activity (sends plus receives) of each node. You should use this performance statistic only with plots of network totals or rates. Total network activity per node is shown as a percentage of the total network activity for the entire network.

EST_TOPOLOGY – Topology Information

The EST_TOPOLOGY probe writes information collected by the TOPOLOGY probe (see below) to the **netmain_srvr** log file.

Default Probe Interval Time: 1:00:00

Default Probe Skip Distance: 1

HW_FAIL – Hardware Failure Messages

The HW_FAIL probe records every change in the hardware failure message reported by the **nodestat** command, on the node that is running the monitor. The "ring hardware failure" message identifies the node that last reported a ring hardware failure. Use the message to locate the problem node or cable in the network. The node that reports the failure is often contiguous to the failure or is experiencing the failure.

Default Probe Interval Time: 0:01:00

Default Probe Skip Distance: Not Applicable

Hardware failure messages are generated by Domain/OS. They appear when there is no network traffic, and Domain/OS is completely unable to send network messages.

MEMORY – Records Counts of Memory Errors on Nodes in the Network

The MEMORY probe lists nodes on which correctable memory errors have occurred.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

NET_SERVICE – Network Service Queue Statistics

The NET_SERVICE probe measures the length of the network service queue backlog on each node. The length is the number of network service requests that remain in the queue immediately after a request is serviced.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

File server, any backlog:
: Shows the number of times that each node's file server noted one or more file service requests in the file server queue. Any requests left in the queue create a queue backlog. The file server checks for a backlog every time it services a request. The file server handles only requests from other nodes for operations such as opening, closing, and creating files (not reading or writing). Output formats show the number of times a backlog was present as a percentage of the number of file services requested by other nodes.

File server, average backlog:
Shows the average number of file service requests in each node's file server queue. Every time it services a request, the file server checks for remaining requests (the queue backlog) and counts these remaining requests. When there are no requests, the performance statistic adds a 0 to the average. The file server handles only requests from other nodes for operations such as opening, closing, and creating files (not reading or writing). Backlog incidence is shown as a percentage of the queue's capacity.

File server, backlog severity:
Shows the average length of each node's file service backlog. Every time it services a request, the file server checks for remaining requests (the queue backlog). If a backlog exists, the number of requests the queue contains are used for averaging in this performance statistic. The file server handles only requests from other nodes, for operations such as opening, closing, and creating files (not reading or writing). Incidence is shown as a percentage of the queue's backlog capacity.

File service queue overflow:
Shows the number of rejected requests for file services for other nodes. A node rejects file service requests when it already has too many other requests pending. Rejections slow down the node that made the request and can cause errors. When a node has service queue overflows, it can indicate that too many other nodes require data stored on this node. Output formats shows this statistic as a percentage of the file service requests made to each node.

Total file service: Shows how many file services each node performs for other nodes (not file services the node performs for its own benefit). You should use this performance statistic only with plots of network totals or rates. File services are activities such as opening and creating files, and directory lookups. (The paging server handles file reads and writes.)

Paging server, any backlog:
Shows the number of times that each node's paging server noted one or more page service requests in the page server queue. Any requests left in the queue create a queue backlog. The paging server checks for a backlog every time it services a request. The paging server handles only requests from other nodes, for reads, writes, paging, and several internal operating system services. Incidence plots show the number of times a backlog was present as a percentage of the number of page services requested by other nodes.

Paging server, average backlog:

Shows the average number of page service requests in each node's page server queue. Every time it services a request, the paging server checks for remaining requests (the queue backlog) and counts these remaining requests. When there are no requests, the performance statistic adds a 0 to the average. The paging server handles only requests from other nodes for reads. Backlog incidence is shown as a percentage of the queue's capacity.

Paging server, backlog severity:

Shows the average length of each node's paging queue backlog. Every time it services a request, the paging server checks for remaining requests (the queue backlog). If a backlog exists, the number of requests the queue contains are used for averaging in this performance statistic. The paging server handles only requests from other nodes, for reads, writes, paging, and several internal operating system services. Incidence is shown as a percentage of the queue's backlog capacity.

Total paging service:

Shows how many paging services each node performs for other nodes (not file services the node performs for its own benefit). You can use this performance statistic only with plots of network totals or rates. Paging services are activities such as reads, writes, normal paging, and some internal operating system services. (The file server handles file opening, file creations, and directory lookups).

Reads requested: Shows the number of pages that each node has read from other nodes in the network. Nodes read pages during file I/O or any other paging activity. Output formats that use percentages show the statistic as a percentage of all the node's reads and writes to other nodes.

Reads serviced: Shows the number of pages on each node that have been read by other nodes in the network. Nodes read pages during file I/O or any other paging activity. Output formats that use percentages show the statistic as a percentage of all services the node performs for other nodes.

Writes requested: Shows the number of pages each node has written to other nodes. Nodes write pages during file I/O, operating system execution, and many other activities. Output formats that use percentages show the statistic as a percentage of all the node's reads and writes to other nodes.

Writes serviced: Shows the number of pages written to each node, by other nodes. Nodes write pages during file I/O, operating system execution, and many other activities. Output formats that use percentages show the statistic as a percentage of all services the node performs for other nodes.

PAGING – Diskless/Partner Information

The PAGING probe records information about diskless nodes and their paging partners.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Diskless nodes/paging partners:
Produces a display of diskless nodes and their paging partners. This display cannot be used with data taken from a running monitor. Use it to analyze data from log files.

Paging partners, ordered by diskless node:
Shows the node(s) used as paging partner(s) by each diskless node.

Paging partners, ordered by mother node:
Shows the diskless node(s) supported by each node that that volunteered as a paging partner.

SWD_10_MSGS – Software Diagnostic Messages (10)

The SWD_10_MSGS records counts for various performance statistics on a node, both before and after a 10–message broadcast. The data collected reflect the effect of receipt of messages on the node's performance statistics.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

SWD ack parity: Counts the number of parity failures in the hardware protocol segments of software diagnostic messages. The hardware always detects ACK parity errors if any occur. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD bus error: Counts errors during Direct Memory Access (DMA) from the ring controller, during receipt of software diagnostic messages. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages received by the node.

SWD CRC:	Counts test messages that contain Cyclical Redundancy Check (CRC), errors, detected by the ring hardware. CRC errors can result from errors in any part of the received message except the hardware protocol segments. Note that this performance statistic shows only those errors that occur in software diagnostic messages. The SWD_10_MSG and SWD_100_MSG probes collects data for this performance statistic. The display formats show the statistic as a percentage of the diagnostic messages received.
SWD end-of-range:	Counts the number of times that one or both of the message fields in the test message received was larger than the Direct Memory Access (DMA) channel allowed for it. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.
SWD header checksum:	Counts SWD (Software Diagnostic) messages that contain checksum errors in the message header. Since the operating system program that verifies header checksums is usually disabled, the count for this statistic should be 0. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.
SWD messages not received:	Shows the percentage of Software Diagnostic messages sent to a node but not received, as a percentage of the total number of SWD messages sent to the node. The SWD_10_MSG and SWD_100_MSG probes collect the data for this performance statistic.
SWD modem err:	Counts the number of times the receiver could not synchronize properly with the network, during receipt of SWD messages. The conditions that cause this error class are biphase or Elastic Store Buffer (ESB) errors. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the SWD test messages the node receives.
SWD overrun:	Counts the number of Direct Memory Access (DMA) overruns that occur during software diagnostic messages. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD packet error: Counts the number of times either the receiver or the transmitter had problems in SWD (Software Diagnostic) messages. If the fault was in the receiver, other counts are incremented also. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD receive timeout:
Counts Software Diagnostic (SWD) messages received that did not complete in the expected time. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages received.

SWD transmit errors:
Counts the number of times that either the transmitter or another receiver had an error in a Software Diagnostic test message. For this error to occur, some other error flag must be set. The SWD_10_MSG and SWD_100_MSGS probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages received by the node.

SWD_100_MSGS - Software Diagnostic Messages (100)

The SWD_100_MSGS records the effect of 100–message broadcasts, in the same manner as SWD_10_MSGS above. This probe collects the same information as SWD_10_MSGS.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

TIME_SKEW - Difference Between Node Clocks

The TIME_SKEW probe detects differences between the node clocks.

Default Probe Interval Time: 3:00:00

Default Probe Skip Distance: 1

Compute offset times:
Automatically computes offset times for each log file. The computed offset times are derived from the contents of the log files, using results from the TIME_SKEW probe. A variety of conditions can prevent the offset computation from working. For instance the TIME_SKEW probe may never have executed or may not have operated on each node.

TOPOLOGY – Total Node List Estimate

The TOPOLOGY probe provides total node count (estimate).

Default Probe Interval Time: 1:00:00

Default Probe Skip Distance: Not Applicable

Topology list from node total:

A monitor keeps an estimate of the complete ring topology. The estimate represents a running total of lcnode results, not just the most recent result. This topology list will probably be longer than the list produced by the lcnode command, especially if the monitor has been running for a long time. In particular, it may list nodes that are not currently running. See also ''EST_TOPOLOGY'' above.

MODEM_ERRS – Transmit Modem Errors

This observer reports on nodes that have more than five times the average number of "Transmit Modem Errors.''

Default Probe Interval Time: 0:30:00

Default Recheck Interval: 12:00:00

WIN_CRC – Disk Drive Errors

This observer reports on nodes that have more than 0.01 percent of Winchester disk drive CRC errors.

Default Probe Interval Time: 0:30:00

Default Recheck Interval: 12:00:00

A.4.3 Using netmain_srvr to Build Topology Lists

The **netmain_srvr** and the **netmain** interactive tool provide several kinds of topology lists for the Apollo Token Ring network, including a Master Topology List. The Master Topology List includes:

- The direction of data flow in the network: the monitor from which information is collected and all nodes downstream of that node
- All nodes whether or not they are currently connected to the network

- Each node's hexadecimal ID
- Diskless nodes

Additional information that should be added to this list includes

- Loop number
- Node type
- Paging partners of diskless nodes
- Network servers running on the node
- Node administrator's name (if they occur at your site)
- Node location

When complete, this list is the network's Master Topology List. The Master Topology List provides the foundation for allocating network resources and for monitoring and "tuning" network performance.

The netmain_srvr Topology Lists

When you run **netmain_srvr**, the server process creates a Total Node List. The list is **netmain_srvr**'s best estimate of what the network topology would look like if every node in the network were switched on. To generate the list, **netmain_srvr**'s TOPOLOGY probe performs the equivalent of the **lcnode** command at scheduled intervals.

Under normal operating conditions, if the TOPOLOGY probe runs for several days to a week, it produces a complete network topology list. Since it is repeated at scheduled intervals, the TOPOLOGY probe may find a node at some probe that it did not not find in previous probes. The node may have been newly installed, its loop may have been switched out during a previous probe, or it may have been powered down or not communicating on the network. The TOPOLOGY probe adds newly acquired node information to **netmain_srvr**'s Total Node List.

The EST_TOPOLOGY probe writes **netmain_srvr**'s Total Node List to the **'node_data/system_logs/net_log** directory. The directory contains logs from all **netmain_srvr** probes and observers. The **netmain_srvr** uses non-ASCII files for its work. You can look at the files in **'node_data/system_logs/net_log** by using the netmain interactive tool. In fact, this tool provides the only means you have of looking at any information in **netmain_srvr's net_log** files. (You can look directly at error logs.)

To generate the Master Topology List, run **netmain_srvr** with its default options. If you use the **netmain** interactive tool on the monitor while you are collecting a topology list, be sure the TOPOLOGY and EST_TOPOLOGY probes remain active. After you have run **netmain_srvr** for several days to a week, enter the **netmain** interactive environment from the shell. Choose the Find Monitors and Nodes menu. Use "F4 Topo list from node total" to cause **netmain_srvr**'s Total Node List to be written to the **netmain** environment. When the topology list status message tells you the list is written, you can use "F6 Display topology list" to display the topology list. Verify that the total of nodes on the list equals the total of nodes in your network. If it does not, let the monitor run longer. Store the topology list on line and post printed output in the network control room. Section A.5 describes how to save displays generated with the **netmain** tool. Add the additional information, listed above, to the topology list after storing the list as an ASCII text file.

The **netmain** tool can also generate a topology list by executing the **lcnode** command once. From the Find Monitors and Nodes menu, use the "F3 Topo list" from **lcnode** command to generate the list. Use the F6 command to display the list. This topology list has the same uses, advantages, and drawbacks as those described for the **lcnode** command.

In large networks, subsets of the Master Topology List are often useful. A subset of the master list might contain, for example, only DN3000 nodes. Different types of nodes perform differently. Special purpose topology lists can readily show the kinds of nodes that are experiencing problems or unexpected patterns of use. Use special–purpose topology lists and performance statistics to improve network performance.

If your network extends through several buildings, use subsets of the Master Topology List to identify all nodes in a single building. Use the list to identify conditions applicable only to that building, for example, to isolate problems caused by interference in power supply.

Finally, you should have a topology list of all nodes running monitors in the network. The F2 command "Find all monitors" will generate this list.

A.4.4 Using netmain_srvr to Gather Performance Statistics

The topography and topology lists provide base–level information about your network. Set up a system of regularly scheduled reports from **netmain_srvr** monitors to build a picture of the network's performance over time.

The **netmain_srvr** is the main source of network performance statistics. Shell commands can provide limited information on the performance of nodes and the network. Use **netmain_srvr** as part of an ongoing network maintenance effort because it is the only means of collecting information about

- Events as they occur over time
- The time at which an event count begins

- An event's pattern of occurrence
- Events that occur regularly, but at a low frequency (for example, 1 in 10,000 times)

In addition to generating topology lists, **netmain_srvr** collects more than 75 different statistics (defined earlier in this section) on node and network performance and error conditions. You control **netmain_srvr** probes and observers with **netmain_srvr** configuration files or the **netmain** interactive tool. Both allow you to specify

- What data is collected
- How often the data is collected

The **netmain** interactive tool also allows you to

- Change specifications on active monitors
- Choose a variety of output formats for the data

(See Section A.5 for more information on the **netmain** tool.)

Relationship of netmain_srvr Probes to Network Topology

Information on the ring originates at a transmitting node or source. The source node sends data by placing a message, or packet, on the ring. The packet contains the destination address of the node that the packet is for.

When it receives a packet, a node examines the packet's destination address. If the destination address matches the node's ID, the node

- Copies the packet into its memory
- Modifies the packet to acknowledge successful (or unsuccessful) receipt
- Sends the modified packet to the next node

When a node receives a packet with a destination address that does not match its own ID, it sends the packet to the next downstream node. When a packet returns to its source node, the source node removes the packet from the ring and examines the packet to determine if it was successfully received.

The **netmain_srvr** probes report on the state of

- The hardware that sends and receives packets, specifically the ring controller and modem
- The packet transmission protocols, the condition of the packets sent and received, the number of packets delayed or lost, and the total number of packets sent and received

A node may be unable to send or receive messages if its ring hardware is not functioning properly. If some part of the route between the source and destination nodes is completely broken, the message will never arrive at the destination node.

A packet can leave a node only if it has the correct format. However, a message can arrive at the destination node in a corrupted state for several reasons. Frequently the cause is a weak signal or intermittent signal interference on the coaxial cable. All nodes cause momentary instability on the ring when they are put into the network. That momentary instability can cause a packet to become corrupted.

Both serious and minor conditions are reflected in network data. A ring hardware failure is serious and requires immediate attention. Other conditions can slow down overall network performance but do not cause complete network failure.

All nodes monitor the condition of packets whether or not the packet is intended for the node. Error conditions indicating hard breaks in the network are detected by using the broken-link detection mechanism. Nodes that are downstream of a break report the failure; source nodes upstream of a break are unable to send messages past the broken link.

In addition to the data described above, other **netmain_srvr** probes report on both normal and potentially problematic performance of storage volumes.

Probes Reporting Error Conditions

The probes named in Table A-1 detect potentially serious node and network error conditions. Any of these errors can happen at random, in short bursts. In these cases, the cause is usually a momentarily weak signal. However, if the counts of these probes show a steady increase, there is reason to suspect a problem with a disk, a node, or network cable.

Table A-1. *netmain_srvr Probes Reporting Serious Error Conditions*

Probe	Level of Error
DISK_ERRS	
CRC errors	Count should not be higher than 0.01%
Seek errors	Count should not be higher than 0.01%
Equipment check	Count should not be higher than 0.01%
ERR_COUNTS	
Receive Header Checksum	Count should be 0
Biphase Error	Can indicate hard break in network
ESB Error	Can indicate hard break in network
Modem Error	Can indicate hard break in network
Transmit No Return	Can indicate hard break in network
HW_FAIL	Indicates hard break in network

Probes Reporting on Network Performance

The **netmain_srvr** probes track the services nodes perform for each other. These are

- Paging service—transferring 1024-byte pages of objects from one node to another
- File service—requests from remote nodes to create, open, or close a file on the local node
- Reading pages on and writing pages to other nodes
- Serving as a paging partner for diskless nodes

Node CPU and disk activity is closely related to the services nodes perform for each other; **netmain_srvr** also collects information on this activity. Table A-2 shows the **netmain_srvr** node and network performance statistics that can be useful for allocating network resources.

Table A-2. netmain_srvr Probes Reporting on Network Performance

Probe	**Event to Monitor**
CPU_TIME	Aegis CPU User CPU
DISK_ERRS	Total Storage Module Activity Total Winchester Disk Activity
NET_SERVICE	File Server Backlog Paging Server Backlog Reads/Writes Requested File Server Queue All Paging Server Queue Reads/Writes Serviced Total File Service Total Paging Service
PAGING	Monitor all events generated by this probe

Controlling netmain_srvr's Data Collection Characteristics

The **netmain_srvr** can collect a large amount of data, but you can limit the number of active probes on each monitor if there is a shortage of disk space in the network or if you simply find it easier to do so. For example, you might configure a monitor to collect only topology data, another to collect network performance statistics, and another to collect error statistics.

Decrease a probe's sampling interval time and skip distance; that is, cause the probes to collect information more often, especially when you suspect a problem with a node, a loop, or the network. Decrease the default schedules if you need greater detail about some network performance events or if your service representative requests the information for network maintenance purposes. The default probe sampling intervals are sufficient for most network data collection purposes.

Use the **netmain** interactive tool's Alter Probe Timing submenu to change sampling intervals and skip distance on an active monitor. In **netmain_srvr** configuration files, **-observer** and **-sample** options control the sampling interval time. The **-skip** option controls the skip distance.

You can explicitly set a limit on the length of the log files written by **netmain_srvr**. Use the **-ll** (log length) option in **netmain_srvr** configuration files. The **netmain** interactive tool's Alter Logging Controls submenu allows you to limit log file size. For regular network data collection, it is good practice to

- Limit the length of log files; the default length is usually adequate, but verify this by regularly inspecting log size when you first start monitors or change monitor parameters.

- Automatically start new log files after you close old log files; this is the default and should not be overridden by the configuration file option **-nl**.

- Use the Alarm Server **-disk** [*n*] option on user nodes running monitors. A user logged on to that node will receive notice of a potential disk overflow condition. The Alarm Server startup is explained in Chapter 3 of this book.

Limiting the number of probes on each monitor, closing the log files, and opening new ones frequently (for example once a week) is usually sufficient to ensure that **netmain_srvr** does not overwrite data because disk space is used up or log length is reached. After starting **netmain_srvr** processes on nodes, check the log files written by the servers on a regular basis, typically once a week. Copy old **netmain_srvr** log files to floppy disk or tape for later analysis and permanent storage.

A.5 The netmain Interactive Tool Reference

This section describes **netmain**, the interactive tool used to manage the network and to diagnose problems with individual nodes or disks. It provides complete reference information on the tool and describes how to invoke and use it.

The **netmain** interactive tool can manage **netmain_srvr** monitors. The **netmain_srvr** monitors collect information used to allocate network resources and diagnose network problems. Most of this sections contains reference information on all **netmain** menus. The contexts in which to use **netmain** are described in Section A.1 of this Appendix.

Refer to this description of **netmain** when you set up a network management program and when you work with the tool. If you have never used **netmain**, practice with the tutorial at the beginning of this Appendix (Procedure A-2). The tutorial does not give detailed information about the menu commands. Its purpose is to allow you to become familiar with controlling the the **netmain** menus. Read the descriptive information in this Section on **netmain** before or after you practice with the tutorial.

You can configure **netmain_srvr** from the command line (see Section A.4), or you can start a server process and use the **netmain** interactive tool to configure it. The menu and submenu descriptions below reference the server options controlled from each menu.

When **netmain_srvr** is running, the **netmain** lets you control **netmain_srvr** monitors, change parameters, and analyze the information in the log files. In addition, **netmain** provides output formats and graphic displays of the information collected by the **netmain_srvr** process. The tool provides online help about menu usage and about every menu item.

A.5.1 Invoking netmain

Use the shell command **ps** (UNIX operating system command) **or pst** (Aegis command) to verify that a monitor is running before you start using **netmain**. The tool can find monitors that have been running for approximately 30 seconds. If the tool cannot locate it, the monitor may be experiencing a problem.

Invoke **netmain** from the shell command line as follows:

```
$ netmain
```

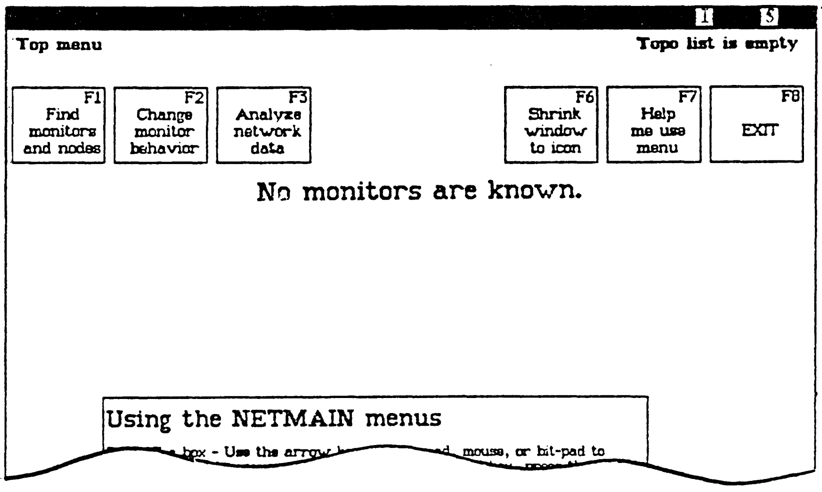
You can run the **netmain** interactive tool in an icon window by selecting the F6 box in the Top Level menu and using the space bar to create the icon. Open the window from its icon mode by placing the cursor in the icon and pressing the space bar once.

When you use **netmain**, select a **netmain_srvr** monitor to work with before using any other feature of the program.

A.5.2 The netmain Top-Level Menu

When you invoke **netmain**, the program displays its Top-Level menu, shown in Figure A-5. The commands on the menu are described in Table A-3. Use the Top-Level menu to

- Invoke the other menus
- Run **netmain** in an icon window



```
Top menu                                   Topo list is empty

                No monitors are known.

Using the NETMAIN menus
box - Use the arrow [illegible] mouse, or bit-pad to
```

Figure A-5. Top-Level Netmain Menu

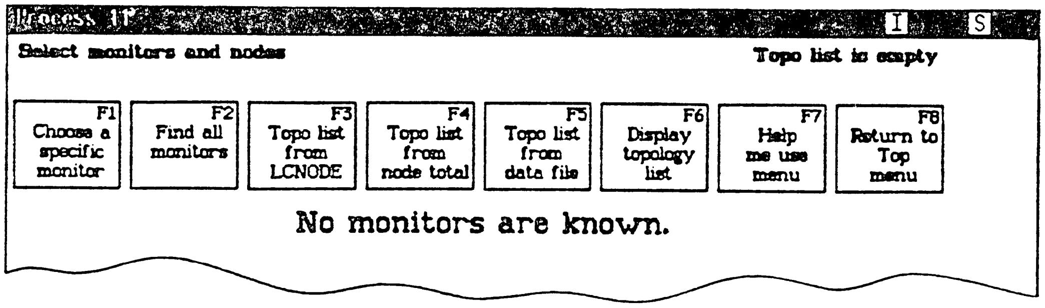
Table A-3. Top-Level Commands

Command Box	Description
F1 - Find monitors and nodes	Invokes a submenu that selects a single monitor from which to collect information or generate reports about monitors or nodes. Use this submenu to select monitors or to create and manipulate network topology lists.
F2 - Change monitor behavior	Invokes a submenu that redefines the **netmain_srvr** parameters. Use this submenu to change the behavior of an active monitor or to close an active log file for examination.
F3 - Analyze network data	Invokes a submenu that formats data and displays it on the screen. You can copy formatted data to a file for printing at a later time. Use this submenu to format data from an executing monitor or from a log file that has been closed.
F6 - Shrink window to icon	Runs the program in an icon window.
F7 - Help me use menu	Provides online help.
F8 - Exit	Quits the program.

A.5.3 The netmain Find Monitors and Nodes Menu

Figure A-6 shows the Find Monitors and Nodes menu. Use the Find Monitors and Nodes menu to

- Find monitors
- Select monitors for use
- View network topology lists


Select monitors and nodes

Topo list is empty

No monitors are known.

Figure A–6. Find Monitors and Nodes Menu

The F1 and F2 command boxes in this menu find specific monitors. The F3 through F6 command boxes write to and display the topology list. Table A–4 explains each command box in the Find Monitors and Nodes menu.

Table A–4. Find Monitors and Nodes Commands

Task	Command Box Used	Explanation
Select a monitor for examination or configuration	F1—Choose a specific monitor	Selects a monitor for work or analysis. Use this option when **netmain** reports "No Monitors are Known," to examine the log file currently open for this monitor; or the monitor itself; or to reconfigure the monitor. Use the input to specify the monitor by following the guidelines in the help box. **Netmain** finds the monitor and draws a box containing that node's name. If it can't find the monitor, **netmain** reports "Node appears not to have a monitor." This can happen if • The monitor is busy and cannot respond • The monitor has stopped running • The network is broken so that messages are not transmitted

Table A-4. Find Monitors and Nodes Commands (Cont.)

Task	Command Box Used	Explanation
		If the monitor is busy, try the command a second time if you receive this message. You cannot use other **netmain** features until you choose a monitor.
List all monitors	F2—Find all monitors	Finds all monitors, using the current topology list (see F6). If the topology list is empty, **netmain** executes **lcnode** and writes the results to the Topo list. Status messages appear as the program searches for monitors. **Netmain** draws a box containing the name of each monitor it finds. Select a monitor by pointing to the box. **Netmain** encloses it with stripes.
Fill topology list with current **lcnode** information	F3— Topo list from **lcnode**	Executes **lcnode** and writes the information to the Topo list. To display the list, use the F6 command box.
Fill topology list with an estimated of all the nodes physically connected to the network — even those not communicating on the network	F4—Topo list from node total	Takes the Total Node list stored in the monitor selected and writes it to the Topo list. Since the Total Node list is a cumulative list of nodes that the monitor found after several **lcnodes**, it should be the most complete online list of all the nodes in the network. When the monitor has run for a week, the Total Node list should include all nodes that have been on the network since the monitor started. The list may be longer than the current **lcnode** report. If you are using the monitor on the local node, **netmain** gets a Total Node list without generating network traffic. Use this command on a

Table A-4. Find Monitors and Nodes Commands (Cont.)

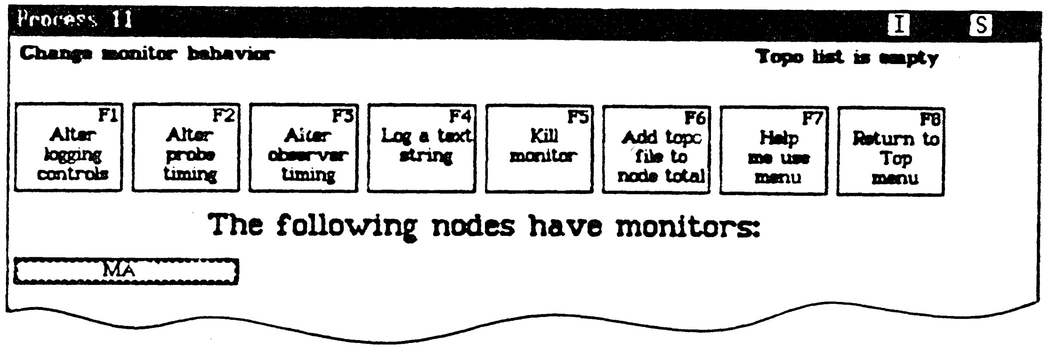
Task	Command Box Used	Explanation
		local monitor when you need to generate a topology list during network problems. To display the topology list, use the F6 command.
Fill topology list from a file	F5—Topo list from data file of nodes	Prompts for a filename and list writes its contents to a topology list. Use a log file or create a text file. If you create a text file, list node names or hexadecimal IDs, separated by spaces, or listed on different lines. Comment lines must start with the characters # or {, and the text must run to the end of the line. Use this command to build a Master Topo list or special purpose lists, for example, a list of nodes that run **netmain_srvr** or any of the network servers. The help box shows how **netmain** resolves relative pathnames. To display the contents of the topology list, use the F6 command box. If you don't enter a filename in response to the prompt, **netmain** displays an error message.
View current network or topology kept in stored topology list.	F6 — Display Topo list	Displays the current contents of the topology list in a read-only pad. You can use other commands (F3, F4, F5) to fill the topology list with particular information. If you have not used one of these other commands yet, **netmain** performs an **lcnode**, fills the topology list with the results, and displays the contents of the list. To save the list, use the DM **pn** command.

A.5.4 The netmain Change Monitor Behavior Menu

The Change Monitor Behavior menu allows you to manipulate monitor behavior. The **netmain_srvr** option **-topo [init]** corresponds to the F6 command in this menu. In order to use this menu, you must select a monitor by using the Find Monitors and Nodes menu. Then you can use this menu to

- Close a current log file so you can analyze its contents
- Schedule probes to run more frequently
- Reschedule observers
- Configure data collection parameters for each monitor

Figure A-7 shows the Change Monitor Behavior menu, and Table A-5 lists each command box in the menu.



Change monitor behavior

Topo list is empty

The following nodes have monitors:

Figure A-7. Change Monitor Behavior Menu

Table A-5. Change Monitor Behavior Commands

Task	Command Box Used	Explanation
Set parameters for log file	F1—Alter logging controls	Invokes as submenu that starts new log files, which allow you to change logging parameters.
Set parameters for probes, list probes	F2—Alter probe timing	Invokes a submenu that reschedules probe active and waiting periods and probe skip distances.

Table A-5. Change Monitor Behavior Commands (Cont.)

Task	Command Box Used	Explanation
Set parameters for observer and list observer	F3—Alter observer timing	Invokes a submenu that reschedules the intervals at which observers wake up to check for conditions.
Send a text message to a monitor	F4—Log a text string	Prompts for a line of text that can contain up to 80 characters and sends the text to the selected monitor. To select a monitor, point to the box that contains its name. **Netmain** encloses the box with stripes. If the monitor name box is not displayed, return to Find Monitors and Nodes menu and choose "F2—Find all monitors".
Stop a monitor	F5—Kill monitor	To stop a monitor from **netmain,** point to the box that contains its name. **Netmain** encloses the box with stripes. If the monitor name box is not displayed, return to a Find Monitors and Nodes menu, choose "F2—Find all monitors". Note that once you stop the monitor, you must restart it.
Update a monitor's node list when the TOPOLOGY probe cannot	F6—Add topo file to node total	Updates the chosen monitor's Total Node from a topology list in a text or log file. Use this feature only if the network failed and the TOPOLOGY probe on a monitor you just started cannot get topology information to update the list. This feature should not be used at other times because, if the topology list used does not include all nodes, the monitor will not collect data from any nodes not in the topology list. The help box shows how **netmain** resolves relative pathnames.

Using the Change Monitor Behavior Menu

When you are using the Change Monitor Behavior menu, do not use CTRL/Q (Quit a process) between the time you select the "Perform changes" commands and the time the message "Node_name acknowledged" appears. Doing so may cause unpredictable monitor behavior.

A.5.5 The netmain Alter Logging Controls Submenu

The Alter Logging Controls submenu displays the parameters in effect for the log file currently in use. There are two versions of this menu, the one displayed depends on the monitors logging behavior. The parameters are the same as the following **netmain_srvr** options.

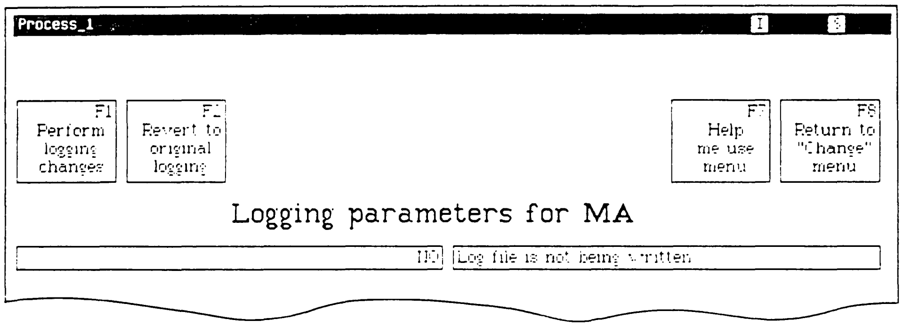
Option	**Meaning**
-l|-nl	Write or do not write to a log file
-l [pathname]	The log file name
-ll n	Limit the size of the log file

Use the Alter Logging Controls submenu to

- Change the parameters in effect for a log file, or
- Cancel changes and return to the parameters most recently in effect

The Alter Logging Controls submenu is shown in Figure A-8. Table A-6 lists each command box in the submenu.

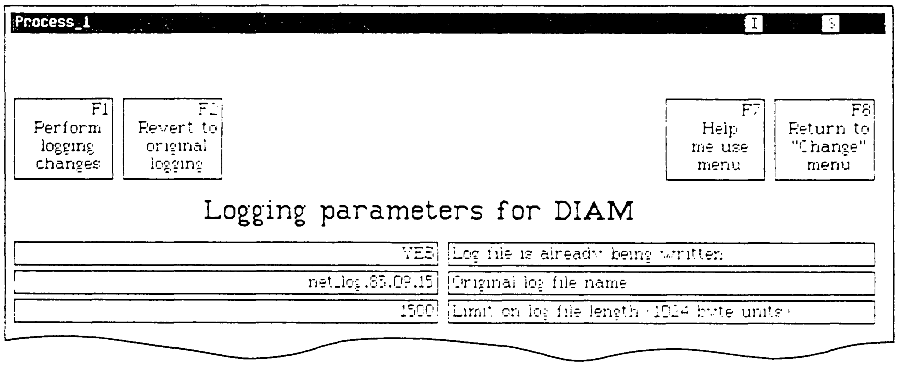
Menu A
No logging


Process_1
F1 Perform logging changes
F2 Revert to original logging
F7 Help me use menu
F8 Return to "Change" menu

Logging parameters for MA

NO | Log file is not being written

Menu B
Logging


Process_1
F1 Perform logging changes
F2 Revert to original logging
F7 Help me use menu
F8 Return to "Change" menu

Logging parameters for DIAM

YES | Log file is already being written
net_log.85.09.15 | Original log file name
1500 | Limit on log file length (1024 byte units)

Figure A–8. Alter Logging Controls Submenu

Table A–6. The netmain Alter Logging Controls Submenu

Task	Command Box Used	Explanation
Change parameters of log files	F1—Perform logging changes	Causes the changes you make to take effect.
Cancel changes to	F2—Revert to original	Abort the changes. When you use the Alter Logging Controls sub–menu, you make changes on a "scratch pad." Select F1 to make the changes. Select F2 to abort the changes. Changes take effect when the status message, "Node_name acknowl–edged" appears.

Table A-6. The netmain Alter Logging Controls Submenu (Cont.)

Task	Command Box Used	Explanation
Get assistance with the menu	F7—Help me use the menu	Gives online help for using the menu.
Exit this menu	F8—Return to "Change" menu	Returns to the Change Monitor Behavior menu.

Using the Alter Logging Controls Submenu

The number of option boxes that you see in the Alter Logging Controls submenu depends on the monitor's current logging behavior. If you start **netmain_srvr** with the default option, log files are always written (unless you stop them), and the submenu displays three option boxes that describe the log file parameters. If the **-nl** option was used at monitor startup, no logging is performed (unless you start logging), and the submenu displays just one option box. Figure A-8 shows the top sections of both menus.

In both versions, the long rectangular left-hand boxes report the status of each parameter. The long rectangular right-hand boxes name or describe the parameter. To change a parameter, you can select either the left-hand or the right-hand box. To start logging, select either box in Menu A. Menu B then appears.

Use the first box in Menu B, "Logfile is already being written", to close a log file. The options at monitor startup determine whether a new log is written. If the **-nl** option was used, no new log file will be written. If the option was not used, a new log file starts once you close the original file. Before closing a file you plan to analyze, note its name; you must enter the name in the Analyze Network Data menu. Use the second box in Menu B, "Original log file name", to change the name of a log file. **Netmain** automatically closes the current file and opens a new log file with the filename you provide or with the default **name net_log.yy.mm.dd**. (Either name is relative to the **'node_data/system_logs/net_log** directory.) If **netmain** creates more than one default log file on the same date, it appends a suffix, using alphabetical order, to the log file names. Three log files created on the same day would be named

net_log.83.08.19 net_log.83.08.19_a net_log.83.08.19_b

When you change the name, the new name appears in the left-hand box, and the message "Logging will be shut down and restarted" appears. If you choose to use the default name, the next message says "Using default name."

Use the bottom box in Menu B, "Limit on log file length", to display the length to which the current log file can grow or to limit the length of the new log file (the one displayed in the middle box). The length is in units of 1 kilobyte (1024 bytes); the default length is 3000 (3 megabytes). You cannot change the limit for a current log file.

Note that if a log file reaches the maximum length and **netmain** needs to write more data, the program starts to write over the oldest data in the log file. If you want to save this oldest data, you should close the file and store it when it approaches its maximum length.

If you attempt to leave the submenu without choosing "F1–Perform logging changes" or "F2–Revert to original logging," an input box prompts "Do you want to perform the changes? [Y | N]." You must choose to perform the change or cancel the change in order to leave the menu.

The netmain Alter Probe Timing Submenu

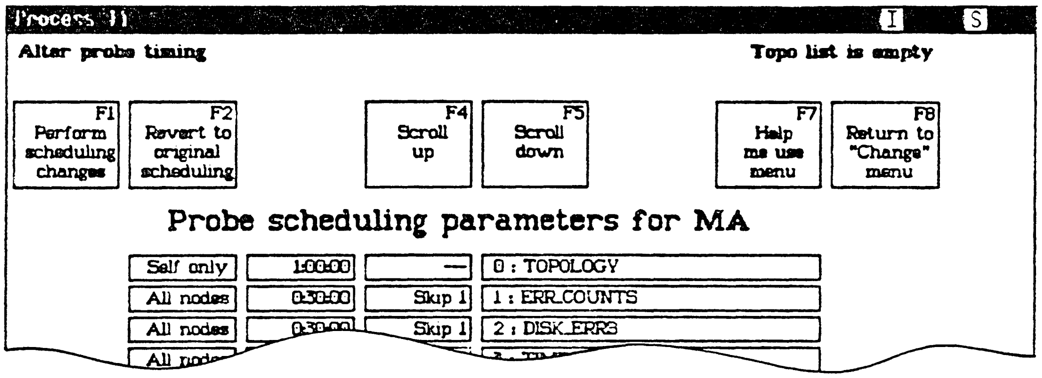
The Alter Probe Timing submenu shown in (Figure A–9) displays the schedule used by each probe when it collects data. These parameters are the same ones as the following **netmain_srvr** options:

–sample	Probe time
–skip	Probe distance

Use the Alter Probe Timing submenu to

- Display the intervals at which probes collect data.
- Change the scheduling intervals. Probes and their default data collection intervals are listed in Section A.4.

Table A–7 lists each command box in the Alter Probe Timing submenu.


Process 11
I
S
F1 Perform scheduling changes
F2 Revert to original scheduling
F4 Scroll up
F5 Scroll down
F7 Help me use menu
F8 Return to "Change" menu

Alter probe timing

Topo list is empty

Probe scheduling parameters for MA

Self only	1:00:00	—	0 : TOPOLOGY
All nodes	0:30:00	Skip 1	1 : ERR_COUNTS
All nodes	0:30:00	Skip 1	2 : DISK_ERRS
All nodes			

Figure A–9. Alter Probe Timing Submenu

Table A–7. The netmain Alter Probe Timing Submenu

Task	Command Box Used	Explanation
Change parameters of log files	F1—Perform scheduling changes	Causes the changes you make to take effect.
Cancel changes to parameters	F2—Revert to original logging	Causes changes made to return last scheduled intervals in effect. When you use the Alter Probe Timing submenu, you're making changes to parameters on a "scratch pad." Select F1 to cause the changes to take effect. If you make changes, then decide not to put them in effect, use F2 before you use F1 to undo the changes in the scratch pad. The changes you make with F1 take effect when the status message "Node_name acknowledged" appears. If you make further changes to parameters and choose F2, the logging parameters revert to the parameter changes that were most recently in effect.
Show probes above those currently displayed	F4—Scroll Up	The menu displays only eight probes at a time. If you are in the lower portion of the menu, use F4 to display the top portion of the menu.
Show probes below those currently displayed	F5—Scroll Down	The menu displays only eight probes at a time. If you are in the upper portion of the menu, use F5 to display the bottom portion of the menu.
Get assistance with the menu	F7—Help me use the menu	Gives online help for using the menu.
Exit this menu	F8—Return to "Change" menu	Returns to the Change Monitor Behavior menu.

Using the Alter Probe Timing Submenu

As shown in Figure A–9, which shows the top [art of the submenu, each probe name appears in the long rectangular box at the right; parameters for the probe are on the left. Table A–8 lists the probe parameters appearing in the leftmost boxes of the Alter Probe Timing submenu.

Table A–8. Probe Parameters

Task	Parameter Box Used	Explanation
Probe scope	First box on left	Indicates whether the probe operates on all nodes or only on the node on which the monitor is running. You cannot change this parameter.
Sampling interval	Second box from left	Indicates how often the probe operates in hh:mm:ss format. The actual probe sampling intervals are slightly longer than the number shown, especially on heavily used nodes. For information about probe defaults, see Section A.4.
Skip Distance	Third box from left	Indicates the distance between nodes checked in each pass around the ring. It is only relevant for probes that check every node. A default skip distance of 1 checks each node on each pass; a skip distance greater than the number of nodes in the ring checks a different node on each pass. For skip distances greater than 1, the probe starts with a different node on each pass. For example, a skip distance of 5 checks the following nodes on each pass:

Table A-8. Probe Parameters (Cont.)

Task	Command Box Used	Explanation
		Nodes 1, 6, ... on the first pass Nodes 2, 7, ... on the second pass Nodes 3, 8, ... on the third pass . . . Nodes 1, 6, ... on the sixth pass Note that the numbers in the above example represent distances from the monitor around the ring. Node 1 runs the monitor, node 2 is the next node downstream, and so on.

If you attempt to leave the submenu without choosing "F1-Perform Scheduling Changes" or "F2-Revert to original scheduling," an input box prompts "Do you want to perform the changes?" [Y | N]; you must choose to perform the change or cancel the change in order to leave the menu.

Guidelines for Scheduling Probes

When you first start to run monitors, you should schedule their probes to collect data very frequently so that you can compile a complete set of data. After several days or a week, you can reschedule the probes.

When you start **netmain_srvr** on a node, leave the TOPOLOGY probe at its default time of 1 hour (01:00:00). Within 48 hours, the log files should include a fairly complete topology list; every node will most likely have been powered on and be online and communicating on the network when the **lcnodes** were run. The **netmain** process then automatically reschedules the topology probe to run every 12 hours to conserve CPU time.

If you change the network topology by adding, removing, or relocating nodes, change the sampling interval to every half hour or so. The **netmain** process will activate the TOPOLOGY probe within 1/2 hour, and the **lcnode** performed by the topology probe will collect the information about new nodes. After this, the topology probe will again run only once every 12 hours.

Note that when a node does not appear in an **lcnode**, the **netmain_srvr** leaves it on the total node list for some time, assuming that it is powered down, that the node's loop is switched out, or that the node is not communicating on the network. If the node disappears for more than a week, the **netmain_srvr** assumes that it is really gone and removes its name from the Total Node list.

Probes that can check each node write a large amount of data to log files. When you configure these probes for each node running a monitor, determine how much disk space you want to allow the log files to consume each day. In general, log file storage requirements

- Increase with the number of probes you use
- Decrease as the sampling interval increases
- Decrease as the skip distance increases

When scheduling probes for your network, remember that log files reach their maximum size most quickly when you check many nodes, sample frequently, and run many different kinds of probes. Sampling less frequently does not fill up the log file quickly, but does not get as much detailed information. If you run fewer probes, you do not fill up the file as fast, but you get less complete information about the network. On any schedule, the longer the recording period, the fuller the log file gets. Make trade-offs based on your needs for network information.

The **check interval** for each probe is the product of the probe's sampling interval and the skip distance. The check interval determines how often the probe on a particular monitor actually checks each node. The following equation shows the check interval.

c = rn s

Where:

c Is the check interval (i.e., how often the probe checks each node)

rn Is the probe n's sampling interval

s Is the skip distance

Using this formula, we could schedule a monitor's ERR_COUNTS probe, for example, to a skip distance of 10 and a sampling interval of 3 minutes (00:03:00). The check interval is 30 minutes, so the ERR_COUNTS probe checks each node once every 30 minutes.

If you are experiencing an ongoing problem with node or network performance or reliability, you should run probes at regular intervals, with more frequent samples and lower skip rates, in order to locate the problem. The smaller the check interval, the more data you will collect and the better your chances for detecting the problem.

When you change probe scheduling parameters, each probe wakes up as soon as it is rescheduled. If you reschedule a number of probes at the same time, avoid "bursts" of CPU activity by setting parameters for a single probe, (use "F1–Perform scheduling changes" to change the parameters) and then setting parameters for the next probe. If you change all parameters at once and then perform the scheduling changes, the probes all wake up at once. For a brief period, these multiple wake-ups can affect performance on the node running the monitor.

The netmain Alter Observer Timing Submenu

The Alter Observer Timing submenu (shown in Figure A-10) displays the schedule used by each observer when it collects data. These parameters are the same ones as the following **netmain_srvr** options:

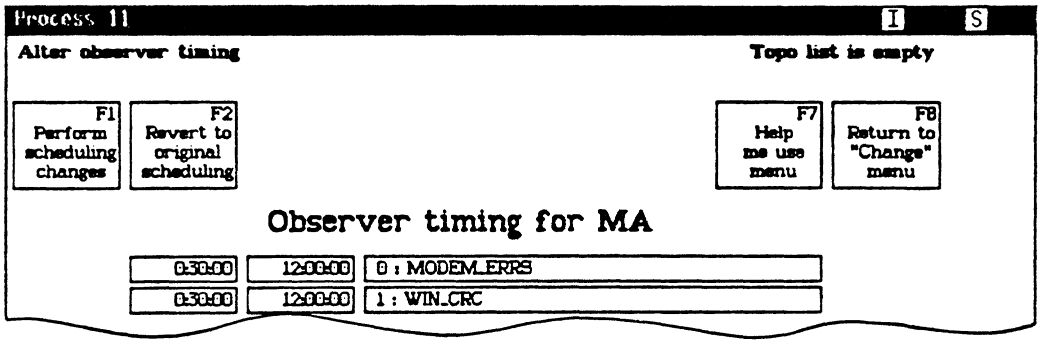
-observe	Observer time
-reobserve	Observer time

Use the Alter Observer Timing submenu to

- Display the intervals at which observers collect data
- Change the scheduling intervals

Observers and their default data collection intervals are listed in Section A.4.

Table A-9 lists each command box in the Alter Observer Timing submenu.



```
0:30:00   12:00:00   0 : MODEM_ERRS
0:30:00   12:00:00   1 : WIN_CRC
```

Figure A-10. Alter Observer Timing Submenu

Table A-9. The netmain Alter Observer Timing Submenu

Task	Command Box Used	Explanation
Change parameters of log files	F1—Perform scheduling changes	Causes the changes you make to take effect.
Cancel changes to parameters	F2—Revert to original scheduling	Causes changes made to return to the last scheduled intervals in effect. When you use the Alter Observer Timing submenu, you're making changes to parameters on a "scratch pad." Select F1 to cause the changes to take effect. If you make changes, then decide not to put them in effect, use F2 *before* you use F1 to undo the changes in the scratch pad. The changes you make with F1 take effect when the status message "Node_name acknowledged" appears. If you make further changes to parameters, then choose F2, the logging parameters revert to the parameter changes that were most recently in effect.
Get assistance with the menu	F7—Help me use the menu	Gives online help using the menu.
Exit this menu	F8—Return to "Change" menu	Returns to the Change Monitor Behavior menu.

Using the Alter Observer Timing Submenu

If you attempt to leave the submenu without choosing "Perform Scheduling Changes" or "Revert to original scheduling," an input box prompts "Do you want to perform the changes?" [Y | N]. You must choose to perform the change or cancel the change in order to leave the menu.

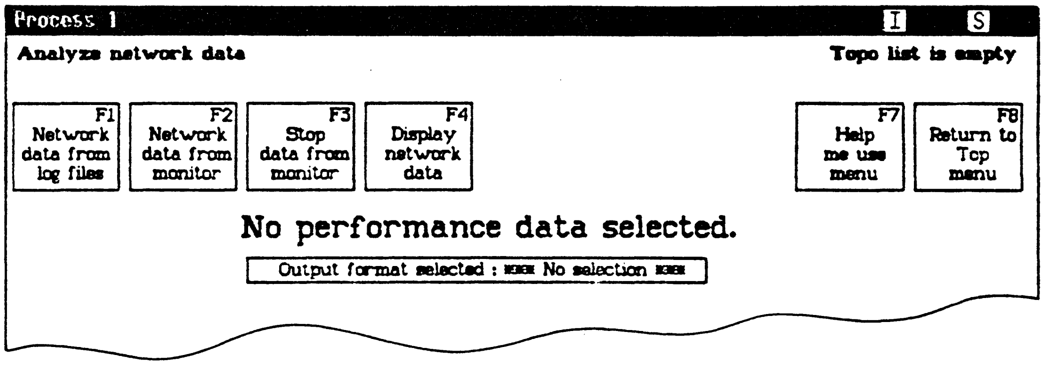
As shown in Figure A-10, the rightmost rectangular box on the menu names the observer, the middle box displays the observer's recheck interval and the leftmost box displays the observer's wake-up interval. The wake-up interval (30 minutes by default) is the interval at which the observer checks the data it has collected. When an observer reports on a node, it does not check data for that node again until after the recheck interval (12 hours by default). A long recheck interval prevents numerous notifications about the same problem.

A.5.6 The netmain Analyze Network Data Menu

Use the Analyze Network Data menu to analyze the information **netmain_srvr** monitors collect about your network. This menu enables you to

- Specify the source of the data you wish to analyze
- Choose a data output format and appropriate parameters for that format
- View the data output format on the display monitor or save the display for printing at a later time

Figure A-11 shows the Analyze Network menu, and Table A-10 lists each command box in the menu.



No performance data selected.

Output format selected : ■■■ No selection ■■■

Figure A-11. Analyze Network Data Menu

Table A-10. The netmain Analyze Network Data Menu

Task	Command Box Used	Explanation
Select a log file	F1—Network Data	Brings up a submenu of log files you can analyze. If multiple log files are specified, **netmain** merges the contents of the files before performing the analysis.
Use data from the current monitor	F2—Network Data from monitor	You must select a monitor with F1 in the Find Monitors and Nodes menu before using this command. Data comes from a running monitor. To get enough data for analysis, use F2 on the Change Monitor Behavior menu to set probe check intervals to short time periods; e.g., 20 seconds. Set the time axis marks to about the same short intervals.
Stop data from monitor	F3—Stop data from monitor	Closes the data stream from a running monitor so it can be examined from **netmain**. After you issue the command, the monitor continues to collect data but does not send data to **netmain**.
Choose a display format output	F4—Display network data	Formats the network data by using the output formats you select. You cannot display data unless selection flags are replaced by some performance statistic or output format. For most displays, you should also choose a topology list, a monitor, and a log file. Data is displayed in a window that can be saved for printing with the DM commands **pn** or **pw**. CTRL/Q halts a display.
Get assistance with the menu	F7—Help me use the menu	Gives online help using the menu.
Exit this menu	F8—Return to Top-Level menu	Returns to Top-Level menu.

Using the Analyze Network Data Menu

Before you can use the Analyze Network Data menu to look at data, you need

- A source of data. Use one or more log files or a running monitor selected with F1 in the Find Monitors and Nodes menu.
- A Topo list. You can use one you've already generated with the Find Monitors and Nodes menu; or, if you don't specify a Topo list, **netmain** will generate one to build a display.
- An Output Format selection.

After you choose a source of data, the Analyze Network Data menu names the log file(s) selected or displays the name of the monitor you are examining. If the log file's pathname is long, **netmain** truncates it.

After you have a Topo list and a source of data, select an output format. **Netmain** displays additional menu items containing parameters for that format. Many parameter menu items have default selections, others display a "**No Selection**" message. When you make a selection or change an existing selection, additional menus or prompt boxes can appear.

After you supply the information necessary, **netmain** collects the data and generates the format you selected.

The following subsections describe what happens when you choose various command boxes from the Analyze Network Data menu.

Selecting the Log Files Submenu

Choosing F1 Select Log Files produces a menu that lets you analyze one or more of the log files you selected with the Find Monitors and Nodes menu. If you choose to analyze multiple log files, **netmain** merges the data from the files.

Merging multiple log files provides more complete information than using a single log file. Several log files may contain data from sequential time periods, for example. Also, if certain loops are switched out of the network and there are monitors in each loop, merging log files merges the data collected by the monitors on each loop. If you didn't merge log files, you simply would not see data for any loops that were switched out.

To select each log file, select one of the long, rectangular boxes in the menu and enter the log file's pathname in the input box that appears. The help information that appears lists the command search rules **netmain** uses if you specify a relative pathname.

If you choose a log file that you haven't closed, an error message appears. Close the file via the Change Monitor Behavior menu or select a working monitor instead.

When you analyze more than one log file, enter the log files in any order. If the log files were produced by monitors on different nodes, correct any differences between the Universal Coordinated Time (UCT) on the nodes running each monitor. If you don't correct these differences, or correct them imprecisely, displays will show numerous "x" marks, representing loss of data.

UCT differences usually exist only in networks that have multiple monitors. Log files copied from one node to another, however, retain the UCT of the node on which they were created.

If you have let the TIME_SKEW probe run, simply select the "F1 Compute offset times" command to make **netmain** correct the time differences by using the TIME_SKEW data. If you use the default sampling interval for this probe, the probe will probably have adequate data to perform the automatic computations. (If it does not have adequate data, the computation will fail.) Using the F1 command with the TIME_SKEW probe is the best offset time correction method.

If you have not let the TIME_SKEW probe run, enter the offset times yourself by using the small box at the right of the box that contains the log file name. Find out the UCT time on each node from which you've chosen a log file. Go to each node, use the **date** shell command, and record the UCT displayed. Be very exact, recording the time it takes to walk between nodes and subtracting it from the UCTs you gather. Then, with the times written down, choose one node as the node with the "base" UCT. Compute the difference (offset) between the base UCT and the UCT on any other nodes from which you've chosen log files. Enter this offset in the form "+hh:mm:ss" by choosing the offset time box and using the input box that appears.

You can also use the **lcnode** command to find the times on nodes that run monitors. If there are many nodes in the network, take into account the amount of time it takes to product the **lcnode** display.

If you use the **calender** utility to reset the UCT on a node running a monitor, there could be an offset between log files created before and after you set the UCT. (Note that time zone settings do not affect the UCT.) To correct for this offset, manually enter the offset between the two times, as just described. Use the old time as a base time, and, for any log file created after the calendar reset, enter the new time as an offset of the old time.

Selecting the Executing Monitors Submenu

Before selecting an executing monitor as the source of data for a **netmain** display, reschedule the probes you want to examine to operate more frequently than the defaults. If you do not reschedule probes, **netmain** cannot produce the display until all nodes have been sampled at least twice. This can take a very long time with default probe scheduling.

A monitor that samples frequently can produce a large amount of data. Ensure that it does not reach its maximum size and begin to write over old data. To provide prompt display of data, take the following steps:

1. Use the Change Monitor Behavior menu to close the current log file. Start a new log file to record data while you examine the active monitor.
2. Reschedule the probe(s) you want to examine to check each node every 15 to 60 seconds.
3. Use the Analyze Network Data menu to select data from the current monitor.
4. From the same menu, set the time axis marks equal to the node check interval. If the node check interval is 00:00:30, set the time axis marks to 00:00:30.
5. Use F4 Display Network Data to produce a real–time display of data from the executing monitor.
6. Once you've finished examining the data from the executing monitor, type CTRL/Q or CTRL/N to stop output to the display.
7. Use "F3 Stop Data from Monitor", and return to the Change Monitor Behavior menu to reschedule the probes to their original sampling schedule. You can start another log file.

A.5.7 Choosing Output Formats for Data

This subsection describes the output formats available for analysis, the performance statistics or other data available in each format, and the parameters to specify for each format. It also provides additional notes about interpreting output in various formats.

Netmain provides graphic display output formats, as well as printed and binary data files. You can view some kinds of data in several different output formats.

Choose a format by selecting "Output format selected" in the Analyze Network Data menu.

- Choose a "density plot" format (gray scale plot) to get a graphic display that shows the behavior of all nodes.
- Choose a "peaks" format (scatter plot) to isolate nodes whose behavior on a performance statistic is above a threshold level.
- Use an "across net" format to survey nodes' relative contributions to network–wide counts for a performance statistic.
- Use the "bar chart" format to look at only a single node's behavior, or to see exact percentages and counts for a performance statistic.

Output Format Descriptions

Table A–11 lists each of the formats and provides a brief description of each. Use the first three formats listed in the table to get information about diskless nodes and network events, for example, hardware failure messages.

Use the other formats listed in the table to display performance statistics collected by **netmain** probes. Performance statistics for the entire network and for individual nodes are available in a number of categories. When you select an output format, a menu of parameters for that format appears. Use the parameter selection menu to choose a category and a performance statistic within that category. (The next subsection, "Output Format Parameters" and Table A–12 describe the parameter menus.)

In most displays, information about a performance statistic appears as a percentage of a total for that category. For example, choosing a performance statistic in the ring transmit category shows occurrences of each node's ring transmit statistics as a percentage of the node's total ring transmits.

"Rate" and "Across net" formats do not show percentages of node totals in a performance statistic category. "Rate" formats, such as "Rate density plot" or "Peak rate" show the absolute number of counts per time unit. In "Across net" formats, such as "Density across net" or "Peaks across net," the percentage shown is each node's contribution to network totals for that statistic.

In the graphs, time intervals appear on the vertical axis. Most of the graphs show data for all nodes in the topology list and show the nodes on the horizontal axis. If the topology list is empty, **netmain** performs an **lcnode** and uses the results.

Table A–11. Output Format Descriptions

Format	Description
Diskless partners	Displays the network's diskless nodes and their partners. Configure the display to show the diskless nodes served by each "mother" disked node, or to show the mother node for each diskless node. Use this format only when you analyze data from log files not from executing monitors.
Scattergram events	Makes a scatter plot of incidences of probe failures or hardware error messages. Hash marks represent event incidences higher than a threshold level set in the parameter menu for this format.

Table A–11. Output Format Descriptions (Cont.)

Format	Description
Bar chart	Makes a bar chart for the performance statistic you specify in the parameter selection menu. Each hash mark (#) in the chart represents incidences of the condition tracked by the statistic, as a percentage of a total for that statistic. For example, a bar chart for a performance statistic in the ring transmit category shows percentages of total ring transmits. Unlike other output formats, you can display a performance statistic for a single node in a bar chart. For a single node, the bar chart shows the statistic as a percentage of a total for that node. If you choose "aggregate," the bar chart shows the statistic as a percentage of a total for the entire network. The bar chart is also the only format that provides exact percentage levels and quantities for the performance statistic chosen.
Peak incidences	Makes a scatter plot of the "peak" incidences of the performance statistic chosen. Hash marks represent a percentage of incidences higher than a threshold you set for the "Top of error scale" parameter in the parameter menu for this format.
Incidence density	Makes a gray scale plot that graphically displays the percentage level for the performance statistic chosen. The percentage level for each node is proportional to the density of the plot. **No color** represents 0%, or a percentage too low to report at the resolution set and a **solid color** represents percentages over the "Top of error scale" number set in the parameter menu. The name for this condition is **saturation.**
Peak rate	Makes a scatter plot that shows nodes that exceed a specific threshold rate (occurrence per time interval) for the statistic chosen. Hash marks represent a rate higher than the threshold set with the "Top of rate scale" parameter in the parameter menu for this format.
Rate density plot	Makes a gray scale plot that shows, for each node, the rate of occurrence of a condition measured by the performance statistic you select. The rate for each node is proportional to the density of the plot. No color represents a rate too low to report at the resolution set, and a solid color represents rates over the "Top of rate scale" set in the parameter menu for this format. The name for this condition is **saturation.**
Peaks across net	Makes a scatter plot that isolates nodes that contribute the highest percentages to network totals for the performance statistic you select. Hash marks indicate nodes whose contribution is higher than the threshold you set with the "Top of error scale" parameter.

Table A-11. Output Format Descriptions (Cont.)

Format	Description
Density across net	Makes a gray scale plot that shows the relative contribution of each node to network-wide totals for the performance statistic chosen. The percentage contribution for each node is proportional to the density of the plot. No color represents a contribution too low to report at the resolution set, and a solid color represents rates over the "Top of error scale" set in the parameter menu for this format.
Printed output	Displays a printed summary of events. You can choose to summarize incidences of hardware failure messages, user messages, probe failures, observer reports, or memory errors.
Binary data file	Dumps data into a binary data file that you can use in specialized network monitoring programs you may wish to write. This format is for advanced users who want statistics not provided by **netmain**. See the insert file **/sys/ins/nud.ins.pas** for the binary file record structure, and see the files **/domain_examples/netmain/nud_example.pas** or **/domain_examples/netmain/nud_example.ftn** for an example of how to use it.

Output Format Parameters

When you select an output format, a menu of appropriate parameters for that format appears. The menu items are identical for many output formats. Table A-12 describes each parameter and shows the output formats for which it is required.

Table A-12. Parameters for Output Formats

Parameter	Required By These Output Formats	Parameter Function
Select a log file	F1—Network data from log files on the Analyze Network Data menu	Brings up a menu of log files that you can analyze. If multiple log files are specified, **netmain** merges the contents of the files before performing the analysis.

Table A–12. Parameters for Output Formats (Cont.)

Parameter	**Required By These Output Formats**	**Parameter Function**
Earliest/Latest data displayed	All, when you use these files for analysis	Lets you limit the time range over which errors are reported, when you're using log files for analysis instead of a monitor.
	None, when you use a monitor	By default, reports start with the earliest data "First in log" and end with the latest "Last in log." To shorten the time range, enter a later time in the prompt box when you choose earliest data displayed, or enter an earlier time in the prompt box when you choose latest data displayed.
Time axis marks	All, except "Printed output" and "Diskless partners"	Lets you change the length of the time units used in formatted data outputs.
Max wait for data	Same as above; also not necessary when you select "bar chart" with actual sampling times	Sets the number of time axis intervals that **netmain** will wait for data from a missing node. After the specified time, **netmain** continues to format data. When you are analyzing data from a log file, you can enter "end–of–file" to read to the end of a log file before continuing with the scattergram. You cannot choose "end–of–file" when you are examining an executing monitor.
Top of error scale	"Bar chart" "Density across net "	Sets the full–scale value or the threshold used in the output format you've chosen. In gray scale plots, incidences higher than the value set cause saturation, so the value you set is a full–scale value. In scatter plots, incidences higher than the value set show as hash marks, so the value you set is a threshold.

Table A-12. Parameters for Output Formats (Cont.)

Parameter	Required By These Output Formats	Parameter Function
		The value set is given in percent units or, for rate plots, as a ratio of counts per time unit). The meaning of the percent unit depends on the format. For output formats that only show levels for performance statistics, the value is percentage of the total that results this condition. For "across net" formats, the value is the percent that each node contributed to the total number of errors of that type over the time interval. The default value is 50%.
Data plotted	"Bar chart" "Density across net "	Specifies the data value you see in plots and scattergrams. Displays a submenu (or peaks) from which you choose one of the categories shown in Table A-11. See Section A.4 for information about specific performance statistics.
Data dumped	"Binary data file"	Specifies data as in "data plotted," above. Here, however, **netmain** sends data to a binary file instead of formatting it for screen display.
Events plotted	"Scattergram events"	Lets you create a scattergram of hardware failure messages or probe failures. See Section A.4 for information about these events.
Data printed	"Printed output"	Lets you choose printed reports on hardware failure messages, probe failures, user messages, or memory errors.
Data ordering	"Diskless partners"	Lets you specify whether the "Diskless partners" format shows diskless nodes for each diskless node (choose "Partners by Diskless".)

Table A-12. Parameters for Output Formats (Cont.)

Parameter	Required By These Output Formats	Parameter Function
Node to plot for	"Bar chart"	Displays an input box that lets you specify the name of the node for which statistics are displayed in a bar chart. By default, the chart shows the sum for all nodes (network aggregate).

Interpreting Bar Chart Displays

The left section of a bar chart contains four columns of information about each time interval. The time and date that each interval ended are in the far left columns. The actual number counted for the performance statistic during this interval, and the number as a percentage of a total for the performance statistic, are in the next two columns. **Netmain** rounds off percentages to the nearest integer value. Each hash mark (#) represents a part of this percentage.

The actual percentage represented by one hash mark depends on on the scale you choose for the graph, using "Top of ________ scale." This value is the full-scale value of the bar graph. With a full-scale value of n percent, the bar graph is at its maximum or full-scale length (50 hash signs) when n percent of the transmit or receive I/O operations result in the error selected. If you use the default value of 50 percent, then the bar graph contains 50 hash signs when 50 percent of the operations result in the error selected. Using the default, then, each hash sign stands for a percentage of 1 percent.

If a node has been rebooted or if a probe has failed because of a transmit failure, the plot reports the reboot or transmit failure during the time interval in which it occurred. A set of greater-than (>) characters to the right of the hash marks indicates that the graph is off scale. Make the "Top of error percentage scale" higher if this problem occurs.

When you choose to plot for a single node, you should set the time axis intervals to correspond to the node's check intervals. This prevents inaccuracies due to the skewing between check intervals and time axis intervals.

When you interpret an aggregate plot, be aware of the limits on the plot's **resolution**. The node check intervals determine the resolution of the data. A check interval of 1 minute will of course produce higher resolution data than a check interval of 30 minutes, but you may be using a check interval of 30 minutes because of log file size limitations. Whatever your check interval, try to set the time axis interval in your aggregate plots to about the same number. Then, when you interpret the plots, note that displays may spread the error incidences out over a period of roughly two check intervals.

For example, if the check interval is 30 minutes and a 2–minute period of network problems occurs, many nodes may not be sampled until up to a half hour after the 2–minute period. Instead of showing a peak for a 2–minute period, the errors could be spread out over a 1–hour period.

If the time axis interval is much longer or much shorter than the check interval, the displays will be skewed even more and will pinpoint peak times for a particular performance statistic less accurately. If the check interval is much *longer* than the time axis interval, the times reported will be very inaccurate.

When a node is rebooted, it resets all its internal counters. The counts for a time axis interval during which a node was rebooted include only those statistics counted after the reset. The counters lose any quantities counted in the time axis interval before the reset. For this reason, time axis intervals which include node rebootings can report an artificially low incidence for the statistic. Charts for single nodes can show "node rebooted" messages in place of a report for that time interval.

Interpreting Scatter and Gray Scale Plot Displays

Scatter plots for hardware or probe failures illustrate incidences of each of these events with an *n* or asterisk (*), where *n* is an integer between 1 and 9, and the asterisk represents any integer greater than 9. Plots for performance statistics show any failure to get information from a node as an *x*.

Peak times shown in scatter plots have the same resolution limitations that they do in bar graphs (discussed in the previous subsection). Also, you should ignore the first line in a scatter plot since it usually shows an incomplete time axis interval.

If your network is very large, with more than 225 nodes, and you attempt to create a plot showing all nodes, the Display Manager (DM) cannot display the rightmost columns in the display for each time interval. To avoid this problem and display all nodes, take the following steps:

1. Take the total node list from the monitor and display it, using the Find Monitors and Nodes menu.
2. Use the DM to cut and paste the list into a new file.
3. Edit the new file to give it a format acceptable to the "Topo list from data file" command. See **netmain** help for details on this command.
4. Now cut and paste part of this new file into a separate file. You should now have two files, each containing a list of half the nodes on the total node list.
5. Invoke the "Topo list from data file" command and name the first file. Then do a plot.

6. Now invoke the "Topo list from data file" command again; this time name the second file. Now do another plot.

7. You now have plots in two windows, each containing half of the nodes. You can line the two windows up by using DM window control commands so that the two windows give the effect of one very large window.

Saving Output Displays

You will often want to save **netmain** displays for future reference. When it makes a display, **netmain** creates a window and a pad, writes the text into the pad using special fonts, and closes the pad. You can save the pad in the fonts used by using the **pn** (pad name) command. Position the cursor in the window, then type the following command:

pn *filename*

Filename must be a file on your node. When you close the window in which **netmain** produced the display, (with CTRL/Y), the *filename* will contain the display in the original fonts. If you just cut and paste to a file, the special fonts will not be used, and the text may be less easy to read.

To print one of these saved files, open the file using <READ> or <EDIT>. Then use the **cpscr** shell command to make a copy of the entire screen display and store it in a second file. Print the file with the **-plot** option. You can use the following command sequence:

cpscr *second_filename*; **prf** *second_filename* **-plot**

See the *SysV, BSD, or Aegis Command Reference* manual for more information about the **prf** and **cpscr** commands.

Index

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing Aegis System Software*
Order No.: 010852-A00
Date of Publication: May, 1988

What type of user are you?

_____ System programmer; language ____________	
_____ Applications programmer; language ____________	
_____ System maintenance person	_____ Manager/Professional
_____ System Administrator	_____ Technical Professional
_____ Student Programmer	_____ Novice
_____ Other	

How often do you use the Domain system?____________

What parts of the manual are especially useful for the job you are doing?____________

What additional information would you like the manual to include?____________

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)____________

Your Name Date

Organization

Street Address

City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

LD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

LD

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing Aegis System Software*
Order No.: 010852–A00
Date of Publication: May, 1988

What type of user are you?

_____ System programmer; language ______________________
_____ Applications programmer; language ___________________
_____ System maintenance person _____ Manager/Professional
_____ System Administrator _____ Technical Professional
_____ Student Programmer _____ Novice
_____ Other

How often do you use the Domain system?______________________________________

What parts of the manual are especially useful for the job you are doing?_______________
____
____
____

What additional information would you like the manual to include?___________________
____
____
____

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)_______________
____
____
____
____
____
____

____
Your Name Date

____
Organization

____
Street Address

____
City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing Aegis System Software*
Order No.: 010852-A00
Date of Publication: May, 1988

What type of user are you?

_____ System programmer; language ______________________

_____ Applications programmer; language ________________

_____ System maintenance person	_____ Manager/Professional
_____ System Administrator	_____ Technical Professional
_____ Student Programmer	_____ Novice
_____ Other	

How often do you use the Domain system?______________________________

What parts of the manual are especially useful for the job you are doing?______________________

What additional information would you like the manual to include?______________________

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)______________________

____
Your Name Date

____
Organization

____
Street Address

____
City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

LD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

OLD