Administering Your DOMAIN System

Order No. 001746
Revision 06

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: December, 1986
Latest Printing: June, 1987

This document was produced using the Interleaf Workstation Publishing Software (WPS), and the InterCAD 2040 Electronic Illustrating System, a product of InterCAD Corporation. Interleaf and WPS are trademarks of Interleaf, Inc.

Preface

Administering Your DOMAIN System is designed to include everything you need to know about system and network administration on DOMAIN networks. We've organized it as follows:

Chapter 1 This chapter is an introduction to system administration.

Chapter 2 "Maintaining Root Directories." This chapter includes procedures to catalog nodes and manage root directories with **ns_helper**.

Chapter 3 This chapter contains a comprehensive discussion of the network environment, including startup files.

Chapter 4 "Creating and Maintaining User Accounts." This chapter contains a description and discussions of the registries and their use in the DOMAIN environment, as well as information on updating and replicating this information.

Chapter 5 This chapter discusses system and software security and the ACL system.

Chapter 6 This chapter contains reference material on **ns_helper** and other servers.

Chapter 7 Chapter 7 describes the information you need to manage the network effectively and how to collect it.

Chapter 8 Chapter 8 discusses methods of managing your network and nodes for optimum performance.

Chapter 9 This chapter describes network troubleshooting procedures.

Chapter 10 This chapter describes **netmain_srvr**, a diagnostic tool you use to manage the network.

Related Manuals

The *DOMAIN System Command Reference* (002547) gives information about using the DOMAIN system and describes the DOMAIN commands found in the */com* directory.

The *DOMAIN System Call Reference* (007196) describes calls to operating system components that are accessible to user programs.

The books *Planning DOMAIN Networks and Internets* (009916) and *Managing DOMAIN Internets* (005694) contain important information about issues you should consider before installing the physical media of a network or internet, as well as information about managing DOMAIN system and software in an internet environment.

The book *Writing Device Drivers with GPIO Calls* (000959) provides information about writing device drivers that the print server (**prsvr**) can support.

Documentation Conventions

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

bold

We use bold type to emphasize keywords in text and command-line examples. A keyword can be

- The name of an executable system object (command or shell script) and any options (switches, regular expressions, or real pathnames) that the command or shell script accepts. For example, **ld -lt**.
- The name of an executable system object (command or shell script) and any options (switches, regular expressions, or real pathnames) that the command or shell script accepts. For example, **ld -lt**.
- The name of a callable function, including all syntactically required punctuation. For example, **open**(*path, flags, mode*).

We do not use bold type for general emphasis. In our ASCII help files, bold type looks the same as roman type.

italics

We use italics to emphasize

- Names or pathnames of system objects. For example, */com/sh* or */tmp*.
- Names we use as stand-ins for names and/or values that you must supply. For example, **prf** *filename*, "...prints *filename* on standard output..."

A command line like **ld** *[options] [files]* indicates that **ld** is a command that can be followed by one or more options and an optional file or files. By extension, this font usage appears in command options and option arguments: **-n***number* means do this function *number* times.

We also use italics to indicate the title of a publication, such as the *DOMAIN System Command Reference*. We do not use italic type for general emphasis. In our ASCII help files, italic type is underlined.

`pica`

Where possible, we use the constant-width pica font (or another "typewriter" style font) in code fragments, and shell or DM scripts. In our ASCII help files, pica type looks the same as roman type.

[brackets] We use brackets to delimit optional command line switches (options) and arguments, as well as to delimit a varying optional token in a pathname, like */sys/node_data[.node_id]*. Brackets are also shell metacharacters that delimit a range or character class.

<angle brackets> We enclose the name of a keyboard key in this type of brackets, for example, <ESC> or <AGAIN>. The < and > symbols are also shell metacharacters used for redirection of input or output. A control function that you execute by pressing the <CTRL> key and the named <KEY> at the same time. For example, <D> sends an End-Of-File.

^<KEY> A control function that you execute by pressing the <CTRL> key and the named <KEY> at the same time. For example, ^D sends an End-Of-File.

<CTRL><KEY> A control function that you execute by pressing the <CTRL> key and the named <KEY> at the same time.

. . . Horizontal ellipses indicate that the preceding item can be repeated an arbitrary number of times. For example, **prf** *file* ... means that you can say **prf** *file1 file2 file3*, and so on.

.
.
. We use vertical ellipses to indicate that an irrelevant portion of text has been omitted from an example.

Note that, when we begin a sentence with the name of a filesystem object, we always capitalize the first letter of the name unless this would result in an ambiguity.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. In order to make it easy for you to communicate with us, we provide the User Change Request (UCR) system for software-related comments, and the Reader's Response form for documentation comments. By using these formal channels you make it easy for us to respond to your comments.

You can get more information about how to submit a UCR by consulting the *DOMAIN System Command Reference*. Refer to the **crucr** (CREATE USER CHANGE REQUEST) shell command description. You can view the same description on-line by typing the following at the appropriate shell prompt

help crucr <RETURN>

For your documentation comments, we've included a Reader's Response form at the back of each manual.

Contents

Chapter 7 Collecting Information about the Network

Illustrations

Tables

Procedures

Chapter 1

System Administration Overview

This chapter provides an overview of the software tasks generally involved in DOMAIN system administration.

This manual refers to all DOMAIN devices that communicate on the network as "nodes" in the network. Actually, each of these nodes might be a device with a display and keyboard, such as a DN3xx, DN3xxx, DN4xx, DN6xx node, or a DOMAIN Server Processor (DSP), like the DSP80 or DSP160. Wherever appropriate, we distinguish between user nodes (with keyboards and displays) and DSPs (without keyboards or displays); for example, since you may use different methods to start a server on a node and on a DSP, we provide separate procedures for these two tasks.

Providing Network Services to System Users

As a system administrator, you are responsible for providing network services to system users. Although you will define the exact scope of services for your site, this manual gives general guidelines for providing standard services. As a system administrator, you generally have the following responsibilities:

- Enable the network naming structure by cataloging nodes with disks and providing for diskless nodes.
- Create servers, which are processes that run on a node and provide access to some service, such as the use of a peripheral device. Chapter 3 describes how to set up servers and also includes information about start-up files. Chapter 6 is a reference chapter on DOMAIN system servers.
- Create and maintain a registry database to define authorized network users. Chapter 4 provides instructions for creating and maintaining a registry.
- Protect the registry and system software, back up user files, and install new software on the network. Chapter 5 provides information about these tasks, as well as a discussion of DOMAIN Access Control Lists (ACLs).

Managing the Network and Nodes

As a system administrator, you may also be responsible for maintaining the physical network and for performing minor hardware upkeep procedures on nodes. (Your service representative performs all major hardware maintenance.) In order to use the network management information in Chapters 7 through 10 of this manual, you should be familiar with your own network's topology, the path traveled by information through the network. To understand your network's topology, you should know the direction of data flow in your network, the nodes' order of sequence in the data path, and the location of cables that connect these nodes.

Chapter 2

Maintaining Root Directories

This chapter applies mainly to installations with a single DOMAIN network. If you have a DOMAIN internet, that is, multiple DOMAIN networks connected by routing service, see *Managing DOMAIN Internets* for additional information about how to catalog nodes and maintain root directories.

Each node has a copy of the root directory that contains the names and hexadecimal IDs for nodes on your network. The root directory provides the associations between the node names and the corresponding ID numbers used by the network. To ensure file access and communications on the network, you must make sure that these directories remain accurate. This section describes the process of cataloging node name — node ID associations in root directories — and of maintaining root directories, and gives step-by-step procedures to carry out these tasks.

Cataloging a Node

The **ctnode** command catalogs a node. That is, it 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 previously cataloged node's name.
- Replace a node's disk.
- Run the **invol** utility.
- Call a service representative to replace the node's ID PROM. The 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 cataloged. We strongly recommend that you give names to all nodes in your network. Otherwise, you leave yourself and all users open to errors; hexadecimal numbers are difficult to remember and easy to type

incorrectly. On networks that do not use **ns_helper** (described below), remote nodes can only access nodes that have been cataloged.

Because each disked node has its own copy of the root directory, cataloging a node is a two-step process.

1. You must first catalog the node in its own root directory (or, for diskless nodes, in the partner's root directory).

2. You must then make this information available to all other root directories.

The procedures for Step 1 are the same for all networks. The procedures for Step 2 depend upon whether you run an **ns_helper** (naming server helper) process on your network. The **ns_helper** is a server process that maintains a master copy of the root directory and provides node name to ID mapping information to nodes. It reduces the cataloging effort when you add nodes or change names. You must run the **ns_helper** process if you have a DOMAIN internet, and you should run it if your network configuration changes frequently. However, you do not need to run **ns_helper** on a small network where

- Root directories usually are current
- Maintaining root directories takes little time
- New nodes are added to the network infrequently

In this case, use the **ctnode** and **uctnode** commands to maintain the root directory.

The following section describes the procedures that catalog a node in its local root directory. The next section describes the procedures for maintaining the network's root directories using **ctnode**. The rest of the chapter describes how to use **ns_helper** on a DOMAIN network. See *Managing DOMAIN Internets* for information about using **ns_helper** on a DOMAIN internet.

Cataloging Nodes In Their Local Root Directories

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 DSP server).
- 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 your Customer Services representative.

PROCEDURE 2-1: Cataloging a Node

1. Log in as **user**.

2. Use the **lcnode -me** command to determine the node's hexadecimal ID. For example:

```
$ lcnode -me
The node ID of this node is 8523.
```

3. Skip this step and go to Step 4 if this is a new diskless node, you changed the disk, or you ran **invol**.

   Use the **uctnode** command to remove (uncatalog) the node's current name. 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.

```
$ uctnode node_8523 -l
 "Node_8523" uncataloged.
```

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.

```
$ ctnode new_name node_id -l
```

     For example:

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

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

```
$ ctnode node_name node_id -l -r
```

5. Update the node's root directory with the names and IDs of all other nodes on the network:

```
$ ctnode -update -l
 3 nodes responded!
 Node 8555 cataloged as "sam_node"
 Node 8525 cataloged as "george_node"
 Node 8523 cataloged as "raster"
```

END OF PROCEDURE 2-1

PROCEDURE 2-2: Cataloging a DOMAIN Server Processor

Use this procedure to catalog DSPs with disks. In new networks, catalog DSPs after you've cataloged nodes with monitors. Get the DSP's node ID from the white inspection slip attached to the shipping carton packing slip. If you do not have the inspection slip, contact your service representative, who will determine the node ID for you; 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.

1. Enter the following command at a node with a monitor to log in to the DSP as **user** (assuming that **user** has a null password):

   ```
   $ crp -on node_specification -login user ''
   ```

   For example:

   ```
   $ crp -on 8523 -login user ''
    Connected to node 8523
   ```

3. Skip this step and go to Step 4 if this is a new diskless node, you changed the disk, or you ran **invol**.

   Use the **uctnode** command to remove (uncatalog) the node's current name. 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:

   ```
   $ uctnode node_8523 -l
    "Node_8523" uncataloged.
   ```

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:

     ```
     $ ctnode new_name node_id -l
     ```

   For example:

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

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

     ```
     $ ctnode node_name node_id -l -r
     ```

5. Update the node's root directory with the names and IDs of all other nodes on the network:

   ```
   $ ctnode -update -l
    3 nodes responded!
    Node 8555 cataloged as "sam_node"
    Node 8525 cataloged as "george_node"
    Node 8523 cataloged as "raster"
   ```

END OF PROCEDURE 2-2

Using ctnode to Catalog Nodes on the Network

Once you catalog a node in its own root directory, you must then update the information in all other nodes' root directories so that the remote nodes can access the newly cataloged node and its files. If the network is small and node configurations change infrequently, use the **ctnode** and **uctnode** commands to manage the network root directories. Procedures 2–3 through 2–6 comprise the steps you must take to update the root directories. Use these procedures as follows:

- Use Procedure 2–1 or 2–2 for each node you are cataloging before using any of these procedures, unless the node's PROM was changed.
- Use Procedure 2–3 if you are creating a new network or adding several nodes to a network.
- Use Procedure 2–4 if you are adding a single node to an existing network.
- Use Procedure 2–5 if you are changing 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 replaces a node's PROM.

PROCEDURE 2–3: Creating a New Network

1. Log in as **user** at any node on the network.
2. Enter the **ctnode -update** command to update the root directory to include 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<RETURN>
2 nodes responded!
Node 8555 cataloged as "sam_node"
Node 8525 cataloged as "george_node"
```

   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 Step 2 until you get a full root directory.

3. Update all other nodes with the information from this root directory. Enter the local node's name in the following command:

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

END OF PROCEDURE 2–3

PROCEDURE 2–4: Cataloging a New Node in an Existing Network

1. Log in to the new node as **user**.
2. Catalog the new node on all other nodes in the network; enter the node's name and ID in the following command:

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

END OF PROCEDURE 2–4

PROCEDURE 2-5: Changing Node Names

1. Repeat the following steps at each node on the network if you change a node's name. (Otherwise, the node will be cataloged under both the new and the old name.)

   a. Log in as **user**.

   b. Enter the **uctnode** command to remove the node's old name from the root directory. For example:

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

2. Log in to any node as **user**.

3. Update the root directories of all nodes in the network; enter the recataloged node's name and ID in the following command:

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

END OF PROCEDURE 2-5

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.

1. Log in to any node as **user**.

2. Update 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 //?*
```

END OF PROCEDURE 2-6

Using ns_helper in Your Network

The **ns_helper** (*/sys/ns/ns_helper*), the Naming Server Helper, is a DOMAIN server process. It provides an automated method of maintaining node root directories. You can use **ns_helper** on any DOMAIN network, but you must use it on each network when two or more DOMAIN networks are joined in a DOMAIN internet. This section describes how to implement and use **ns_helper**.

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.

On large networks and on DOMAIN internets you can have more than one **ns_helper** process, each with its own 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	Consists of the network number (0 if you have only a single DOMAIN network) and the hexadecimal node_ID contained in a node's PROM.
Entry Type	The type of object being cataloged: for disked nodes, system directory (sdir); for diskless nodes, node.
UID	The Unique Identifier for a node's entry directory. If a node is diskless, **ns_helper** assigns a UID.
Entry Date/Time	The date and time at which this entry was made to the master root directory.
Creating Node	The hexadecimal ID of the **ns_helper** node at which this entry was made to the master root directory, i.e., the **ns_helper** that put this information in the database.
	Replica List
Replica List	The hexadecimal IDs of all **ns_helper** nodes. (See "Replicated **ns_helpers,**" later in this chapter.

When to Use ns_helper in Your Network

Use **ns_helper** when

- Node root directories get outdated rapidly and must often be updated using **ctnode -update**.
- You want to maintain root directories from a single location.
- Many nodes in the network have multiple users who are not responsible for node management and updates.
- New nodes are added to the network regularly.
- You have a DOMAIN internet. In this case, see *Managing a DOMAIN Internet*.

Replicated ns_helpers

In many networks, particularly in small ones, **ns_helper** running on a single node provides reliable access to network objects. In some environments, however, you should run the **ns_helper** server on more than one node. Consider running more than one **ns_helper** process when

- There are many nodes in your network. A single server may not be adequate to handle the traffic.
- You want to ensure the reliability and availability of the server process at all times. Two or more servers can provide this insurance.
- There are loops in your network that are switched out regularly and/or your network runs through several buildings. You might want servers in each loop or building.

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**. Therefore, the processes and databases are often called replicas. 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, or propagates, the new information to all the other nodes on the list. When **ns_helpers** receive new information from another server, they update their 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 a few days. In exceptional circumstances of node, loop, or disk failure, 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.

Deciding on the Number and Placement of ns_helper

Your network's topography and topology will influence the number and location of servers. Use the following statements as guides only.

- You must run at least one **ns_helper** server on each network on a DOMAIN internet.
- Run at least one **ns_helper** server for every 200 nodes in your network.
- If your network extends over several buildings, place at least one server process in each building.
- If there are loops in your network that are switched out regularly, place a server inside and outside the loop.

Managing Root Directories with ns_helper

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

When the naming server tries to locate an object, it first looks in the node's local cache of entry directory names. If the name is not in the node's cache, the naming server refers to **ns_helper**'s master root directory for information about the entry name. Also, if the naming server cannot locate an object using the locally cached information, it refers to the master root directory for more recent information about the name. Whenever the naming server gets information from the master root directory, it adds (updates) that information to the local node's cache.

When you use **ns_helper** on the network, you do not need to use the **ctnode** command to maintain a node's root directory except in the following unusual cases:

- When you change the name of a node (disked or diskless) in the **ns_helper** database, the old name is not deleted from the local caches. When you refer to the node by its new name, the naming server updates the local cache. This means that both its old and new names are then in the cache. Use **uctnode** at each node on the network to remove the old entry from all local caches.
- If you interchange the names of two nodes in the **ns_helper** database, you must use **uctnode** locally at each node on the network to remove the old entries. When the naming server refers to the new name, it updates the local cache with current information.

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

The edns Utility

The **edns** utility enables you to manage the **ns_helper** and its database. Table 2–2 briefly describes the **edns** commands. The *DOMAIN System Command Reference* describes the **edns** commands in greater detail. Online help is available by typing

```
$ help edns
```

and

```
$ help edns commands
```

Table 2-2. 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 the DOMAIN 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** as a base.
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.

Synchronizing Clocks on Replicated Databases

The **ns_helper** processes keep only the most recent information about an entry in their databases. The servers use their 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 so that they refer to a single time standard. Check the node clocks periodically and reset them if they diverge by more than a few minutes. **Edns** provides a means of checking 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** at a time when network traffic is light and all nodes are connected to the network. At these times, most nodes can respond to requests for information about themselves from **ns_helper**.

edns and Diskless Nodes

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

diskless_$nnnnn

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

diskless_$0003d3

Edns associates a UID that it generates for a diskless node with the name it gives to the diskless node. This information about the diskless node appears in the master root directory and can be copied to a node's local cache. If you use **ns_helper** in a single DOMAIN network, you should know that the UID and other information that **edns** generates for the diskless node serves as a place marker for information that is used in a DOMAIN internet; it has no effect in a single network.

As a general rule, you should give diskless nodes non-default names. These names can be mnemonic and enable you to refer to the nodes in commands that take node specifications without using the node's hexadecimal ID. However, remember that the diskless node's name is not the name of an entry directory and cannot be used in a pathname argument.

To name a new diskless node on an existing network, simply use the **ctnode -root** command described in the next section, or use Procedure 2-10. 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. For example:

```
$ ld -ln //
joes_diskless       diskless_$003476    nodisk              barebones
calamity            bridge1             comserver
```

The following commands tell you if a node is diskless:

$ **netstat -n** node_spec

$ **lusr -n** node_spec

$ **bldt -n** node_spec

$ **lcnode -me**

For example:

```
$ lusr -n //comserver
 b_jones          *** diskless //comserver ***  partner node: //bigdisk
```

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. On some secure networks where only the system administrator can use the **edns** command, any user can run these commands to manage the master root directory entries for a node.

- Use the following command to delete the entry for a node name in the local cache and the master root directory. If you remove a node from the network, you can use this command at any node to remove 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 or interchange the names of two nodes, all users should also use **uctnode** to delete the old entry or entries from their local root directory.

- Use the following command to add a node name in the local cache 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 local cache 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 reuse an existing name on a new node.

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

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

  ```
  $ ld -root
  ```

System Administrator Procedures for ns_helper

The rest of this chapter contains procedures for managing **ns_helper** processes and databases. Table 2–3 lists alphabetically the situations where you manage the **ns_helper** and its database and indicates the number of the procedure required to do the tasks. In secure networks, the ACLs might be set so that only the system administrator (%.%.sys_admin.%) can use the **edns** command. In this case, only the system administrator can use procedures 2–9 through 2–16. In this case, other users can run the **ctnode** and **uctnode** commands to manage their node's entry in the master root directory as described in the preceding section "User Procedures for Updating the Master Root Directory."

The uctnode command and ns_helper

The **uctnode** command removes one or more specified entry directory names from the local copy of the network root directory. 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.

In networks running **ns_helper**, system administrators who suspect that the naming database on a node contains outdated information often use the **uctnode** command with wildcarding to uncatalog everything in the node's local network root directory, as follows:

```
$ uctnode ?*
$ ctnode this_node
```

This wholesale uncatalog forces the local naming server to get new information about nodes in the network from the master root directory and rebuild the local root directory with up-to-date information.

In earlier releases, the **uctnode** command verified the node name entry in a network root directory with the corresponding node in the network and removed the name from the root directory if the information was incorrect. After SR9.5, **uctnode** no longer queries nodes on the network before it removes names from a root directory; it simply removes the names.

The network root directory on each node is reserved for the entry directory names of other nodes, and it should never contain references to other kinds of objects. The change in **uctnode** operation makes it important to adhere to this rule. If someone executes **uctnode** *?** on your network root directory, and your root directory contains a file or directory that refers to an object other than a node entry directory, **uctnode** will remove the reference to that object from the directory and you will no longer have access to that object by name.

Table 2-3. ns_helper Procedures

Purpose	Procedure
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 the synchronization of clocks on nodes running **ns_helper**	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 do by using the **ctnode** and **uctnode** commands as described in the preceding section "User Procedures for Updating the Master Root Directory."

PROCEDURE 2-7: Synchronizing Node Hardware Clocks

Use this procedure to correct node hardware clocks that are out of synchronization. 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.

1. Make sure you have an accurate timepiece as a standard time reference.
2. Put the node's NORMAL/SERVICE switch on SERVICE.
3. If the node has a display, use the DM **shut** command to 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. Follow the procedures for your model to shut down the node and display the mnemonic debugger prompt.
4. 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 1986/07/11 13:52:03 EDT.
Would you like to reset it? Y

Please enter today's date(year/month/day): 1986/07/11
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. Therefore, do not allow any objects to be created on the node for the length of time equal to the length of time by which you set the clock back. For example, if you set the clock back by one hour, you must wait one hour before you create objects on that node. You can accomplish this by waiting the setback period before you do Step 5.
5. Return the service switch to the NORMAL position and reboot. On a node with a display, type

   > **RE**<RETURN>
   > <RETURN>
   > **EX AEGIS**<RETURN>
6. Using the same timepiece you used in the previous step, repeat Steps 2 through 5 at each of the nodes you selected to run **ns_helper**.

END OF PROCEDURE 2-7

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.

1. If more than one node runs (or will run) **ns_helpers**, check to make sure that the nodes' clocks are synchronized 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**. For example:

```
$ netstat -n //george_node //martha_node
     The net_id.node_ID of this node is 0.5678.
**** Node 4567 ****  //george_node
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 ****  //martha_node
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 reported times are not within one minute of each other, use Procedure 2-7 to synchronize the hardware clocks on the **ns_helper** nodes.

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

3. Edit the */sys/node_data/startup*[.type] script for the node type. Insert the line

   cps /sys/ns/ns_helper

4. Start the **ns_helper** process.

   - If you are working at the **ns_helper** node, 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_specification **-cps /sys/ns/ns_helper**

5. Verify that the server process is running on the node; enter the **pst** command. For example:

```
$ pst
------------------------------------------------------------
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
.
.
```

6. Repeat Steps 2 through 5 at any other nodes that will run **ns_helpers**.

END OF PROCEDURE 2-8

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: martha_node (ID 8521), and george_node (ID 8525), and with six nodes in the network: martha_node, george_node, fred_node, maxine_node, sam_node, and thomas_node.

1. 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.

2. Use Procedure 2–8 to start **ns_helper** on each helper node.

3. Invoke **edns** from any node in the network. Select a node that will run **ns_helper**. Our example selects george_node; its **ns_helper** process becomes the default **ns_helper**. Enter

   ```
   $ edns node_id
   ```

   The **edns** prompt, <edns>, appears. For example:

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

4. Use the **edns init** command to initialize the **ns_helper** database on the current default node. For example:

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

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

   ```
   <edns> ld
   fred_node     george_node    martha_node
   maxine_node   sam_node       thomas_node
   6 entries listed.
   ```

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

6. Use the following command to set the default server process to a replica **ns_helper** node:

   ```
   <edns> set replica_node_id
   ```

   For example, to set the default server to the process running on martha_node, enter

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

Procedure 2-9 (Cont.)

7. Use the following command to initialize the replica database from the first node:

   ```
   <edns> init -from node_id
   ```

   For example, to initialize the database from george_node, enter

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

8. Use the **edns diff** command to verify that the original and replica **ns_helper** databases are identical. For example:

   ```
   <edns> diff martha_node george_node
   The two directories are identical
   The two replica lists are identical
   ```

   If the databases are not consistent, repeat Step 7.

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

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

    ```
    <edns> quit
    $
    ```

END OF PROCEDURE 2-9

PROCEDURE 2-10: Adding Nodes to the ns_helper Master Root Directory

Use this procedure when you add a node to the network. Also use this procedure to give a diskless node a name 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.

1. From any node, enter the **edns** command. For example:

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

2. 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 casey 7206
<edns> add filly 128c
<edns> add catherine_the_great 3333
<edns> add dr_paul 4b70
```

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

```
<edns> ld -n
```

   For example:

```
<edns> ld -n

nodeid name
  7206 casey
  3333 catherine_the_great
  4B70 dr_paul
  128c filly
  503F fred_node
  8525 george_node
  8521 martha_node
  1C68 maxine_node
   5F6 sam_node
  70DE thomas_node

10 entries listed.
```

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

END OF PROCEDURE 2-10

PROCEDURE 2–11: Deleting Names from the Master Root Directory

Use this procedure if you remove a node from the network or to delete any entries that you added incorrectly to the master root directory.

1. From any node, enter the **edns** command. For example:

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

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

```
<edns> del node_name
```

   For example:

```
<edns> del casey
<edns> del filly
```

3. Changes to the database take effect immediately. Use the **edns ld** command to list the directory to be certain you did not make errors.

   For example:

```
<edns> ld
catherine_the_great  dr_paul      fred_node
george_node          martha_node  maxine_node
sam_node             thomas_node

8 entries listed.
```

END OF PROCEDURE 2–11

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 `catherine_the_great` to `cath_g`. 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.)

1. Repeat the following steps at each node on the network if you change a node's name. (Otherwise, the node will be cataloged under both the new and the old name.)

   a. Log in as **user**.

   b. Enter the **uctnode** command to remove the node's old name from the root directory. For example:

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

2. Log in to any node as **user**.

3. Enter the **edns** command. For example:

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

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

```
<edns> del node_name
```

   For example:

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

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

```
<edns> add node_name node_id
```

   For example:

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

END OF PROCEDURE 2-12

PROCEDURE 2-13: Replacing Information 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 (see the *DOMAIN System Command Reference*). These operations replace the node's entry directory UID with a new UID, and you must put the new information in the database.

1. From any node, enter the **edns** command. For example:

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

2. Use the following command to replace the information for the node:

```
<edns> rep node_name node_id
```

   For example:

```
<edns> rep thomas_node 70de
```

END OF PROCEDURE 2-13

PROCEDURE 2-14: Adding or Initializing an ns_helper Replica

Use this procedure when you add an **ns_helper** replica. Use Steps 2 through 5 to reinitialize the database of an existing **ns_helper** replica. In the following procedure's examples, we initialize the new **ns_helper** replica on node thomas_node, node_id 70de, from george_node, node_id 8525.

1. Use Procedure 2-8 to start **ns_helper** on the new node.

2. At any node, enter the following command to invoke **edns** and set it to the new replica node:

   ```
   $ edns replica_node_id
   ```

   For example:

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

3. Use the following **edns** command to initialize the replica database from an existing **ns_helper** replica:

   ```
   <edns> init -from replica_node_id
   ```

   For example:

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

4. 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 original_replica_node_name new_replica_node_name
   ```

   For example:

   ```
   <edns> diff thomas_node george_node
   The two directories are identical
   The two replica lists are identical
   ```

5. If the databases are not consistent, repeat Step 3.

END OF PROCEDURE 2-14

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 a **ns_helper** node with a slow clock being deleted and replaced by older (and inaccurate) data from a replica node with a fast clock. It is recommended that you keep replica node clocks within one minute. **Ns_helper** allows more than one minute of divergence and will indicate the skew only 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, do the following:

1. Invoke the **edns** utility. For example:

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

2. 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
```

3. 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.

END OF PROCEDURE 2-15

PROCEDURE 2-16: Maintaining Consistency of Replicated Databases

Task 1: Comparing Replica Databases

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.

1. To use the command, enter

   <edns> **diff** node_specification1 node_specification2

   For example:

```
<edns> diff martha_node george_node

                value in          value in
                0.008521          0.008525
diff           directory         directory        name

name          name found      name not found      maxine_node
 uid  21089D87.4000503f  2108af41.4000503f       fred_node

 The two replica lists are identical
```

   In this example, **diff** shows that maxine_node is cataloged in martha_node's master root directory but not in george_node's. It also shows that fred_node 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 martha_node, george_node, and thomas_node, the following two commands compare all replicas:

```
<edns> diff martha_node george_node

<edns> diff martha_node thomas_node
```

Task 2: Unifying Replica Databases

The **edns** utility provides two techniques 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 george_node's database any information that is present in martha_node's database but absent from george_node's. It also replaces any information in george_node's database that is older than information about the same entry in martha_node's database. It is important to note that nothing is changed in the "-from" replica (e.g., martha_node's) database. Type

  <edns> **merge** george_node **-from** martha_node

  Since **merge** only operates on the target replica database, it is often appropriate to do a pairwise **merge** to bring two replicas into a consistent state. You would choose a pairwise merge if each database has some information that is missing from the other replica. For example, type

```
<edns> merge george_node -from martha_node<return>
<edns> merge martha_node -from george_node<return>
```

Procedure 2-16 (Cont.)

- 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 using **merge** all. The following procedure uses **merge_all** to ensure that all databases are consistent:

  1. Invoke the **edns** utility. For example:

     ```
     $ 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. For example:

     ```
     <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 `thomas_node` (`node_id 70de`). Enter the following command to add the node:

     ```
     <edns> addrep thomas_node
     ```

  4. Merge all replica databases; enter

     ```
     <edns> merge_all
     ```

     In this case, the default replica node (`martha_node`) is the source node, and this command merges the information from `george_node` and `thomas_node` into `martha_node`. It then merges `martha_node` into `george_node` and `thomas_node`.

END OF PROCEDURE 2-16

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.

1. From any node, enter the **edns** command. For example:

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

2. Delete the replica; enter the **delrep** command. For example:

```
<edns> delrep thomas_node
```

   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.

3. Check 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_specification
```

   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 Step 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. For example:

```
$ edns 70de<RETURN>
The default ns_helper is 0.70de
<edns> info<RETURN>
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. Enter

```
<edns> shut replica_node_id
```

   For example:

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

4. When the process has stopped, remove the following line:

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

   from *'node_data/startup*[.type].

END OF PROCEDURE 2-17

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 if you are deleting this replica.

1. From any node, enter the **edns** command. For example:

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

2. Shut down the replica **ns_helper**; enter

```
<edns> shut replica_node_id
```

For example:

```
<edns> shut thomas_node
$
```

END OF PROCEDURE 2-18

Chapter 3

The Network Environment

This chapter describes the network environment, including information about

- Node specifications
- The network naming structure, including information on such elements as entry directories and variant links
- The individual node's directory structure, including information on system software directories
- How to provide network services
- Node activity at startup and login, including information on start-up and log-in files
- How to administer diskless nodes

Node Specifications

Node specifications identify particular nodes on the network. Node specifications permit a node's communications software to locate other nodes. Typically, you use node specifications with DOMAIN shell commands that allow an **-n** *node_spec* option. The **lusr**, **netstat**, and **lvolfs** shell commands accept node specifications. A node specification can be either a hexadecimal **node ID** or an ASCII **node name.**

Every node has a unique hexadecimal ID number, the node ID, in a PROM (Programmable Read-Only Memory). You can specify a node by means of its node ID. However, since hexadecimal numbers are inconvenient for people to remember, you can also use a node name to specify a node.

Node IDs

You assign node names by **cataloging** the nodes. You catalog nodes with the shell command **ctnode** (CATALOG NODE). Chapter 2 describes the procedures for cataloging and managing node names.

If your installation has two or more DOMAIN networks connected through routing services to make an **internet**, the nodes have a network number that is a prefix to the node ID. The network number for a DOMAIN network that is not joined to other DOMAIN networks is, by default, 0. If your site has only a single DOMAIN network, you need not be concerned with network numbers and internet addresses. Use the node ID with shell commands that take the **-n** option. *Managing DOMAIN Internets* explains how you specify nodes in a DOMAIN internet.

Node Names: Disked Nodes and Diskless Nodes

You can assign node names to both disked and diskless nodes. On disked nodes, the node name serves a dual purpose: it identifies both the physical node and the top-level (node entry) directory on the node's naming (file) structure. (We describe the the naming structure in the next section.) Because diskless nodes do not have their own storage media, their node names only identify the nodes. They do not correspond to entry directory names and cannot be used to access files. Instead, the pathnames to objects created by diskless nodes begin with the entry directory names of the nodes where the objects are stored.

As a result, you get an error message if you use a diskless node's node name with any command that takes pathnames as arguments. For example, if the node named casey is diskless, the **ld** command

```
$ ld //casey
```

has no meaning and results in the message

```
//casey not found
```

You can see the difference between the names of disked and diskless nodes in the output of the **ld** command:

```
$ ld // -st
 Directory "//":

  sys
  type    name

  node    sprout
  node    radish
  sdir    snow_pea
```

The sys type (system type) of a diskless node name is "node." The system type of a disked node entry directory is "sdir" (system directory). "Administering Diskless Nodes", later in this chapter, describes the procedures you use to provide partners for diskless nodes and to give them node name specifications.

The Network Naming Structure

The operating system expects information on nodes to be organized in a hierarchical tree structure called the **naming tree.** Two naming tree components, directories (analogous to tree branches) and files (analogous to leaves on the branches), form text-string names called **pathnames.** You manipulate objects (directories and files) through their pathnames. Figure 3-1 shows the network naming tree, including some of the standard software directories.

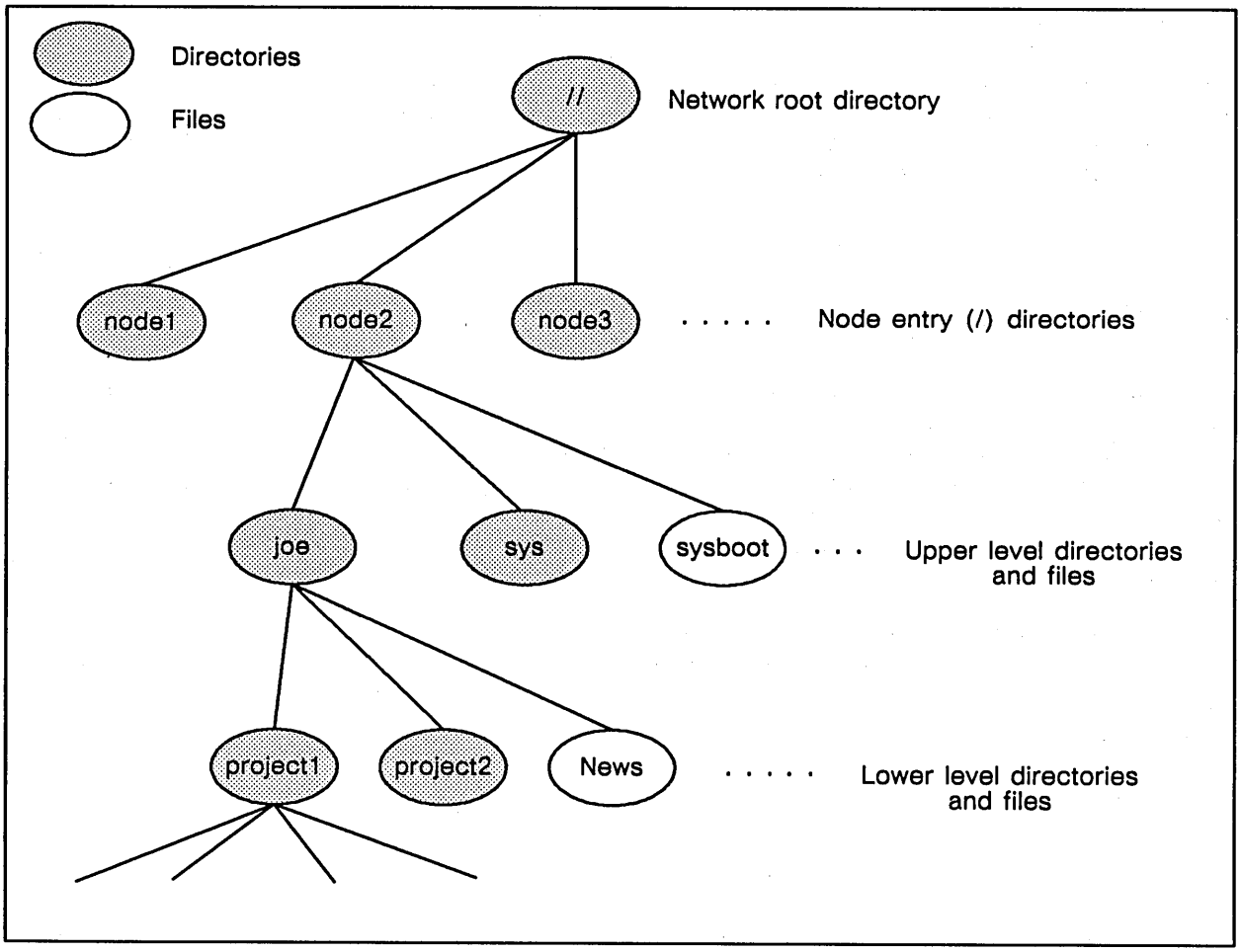


Figure 3–1. Directory Structure

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 also the disked node's name, as described in the previous section.

The set of node entry directory names in your network is the **network root directory**, sometimes referred to as the "//" directory, or 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.

Upper-Level Directories

Upper-level directories are one level below the node entry directory in the DOMAIN naming tree structure. Upper-level directories contain the DOMAIN system software, system and programming libraries, and users' home directories. The system software installation procedures create the system-required upper-level directories. In addition, you can establish upper-level directories on each node, such as home directories for different users.

You should protect the upper-level directories from unauthorized use by creating registries. (See Chapter 4.) You also must protect system software to maintain the node's ability to operate. Software protection is discussed in Chapter 5.

Disk Volumes and Volume Entry Directories

Disked nodes have one or more **physical volumes**, that is, physical media, mounted at any time. A disk unit accommodates a single physical volume; therefore, a node can have as many physical volumes as it has drive units. Each physical volume can be divided into multiple **logical volumes**, logically independent partitions of the physical volume. However, most nodes have one logical volume per physical volume. (This results in the most efficient use of disk space by reducing storage overhead.) You use the DOMAIN **invol** utility to create and maintain logical volumes on the physical media.

Each logical volume that is mounted on a node has a **volume entry directory**. This directory is the logical volume's highest-level directory. The boot volume's volume entry directory is also the node entry directory. A volume's entry directory must be cataloged in the next higher directory in the naming structure. The boot volume (node) entry directory is cataloged in the root directory, the highest-level directory in the network naming structure.

Note that, because the name of a directory is cataloged in the next higher-level directory, you can change the name of a volume entry directory each time you mount the volume. This is particularly useful when you are mounting floppy disks, as you might want to put floppy disks with different software at different locations in the naming tree. Thus, you might mount a floppy disk with data from an analysis of the performance of the node `widgeta` as */tests/data/widgeta*. When you are done using this disk, you could dismount it and mount a disk with financial analysis results as */finances/q286*.

Figure 3-2 illustrates the node `gudrun` with two storage module drives. One physical storage module volume is a single logical volume. This volume is also the boot volume. The second storage module is partitioned into two logical volumes, with entry directories */sm1/vol1* and */sm1/vol2*.

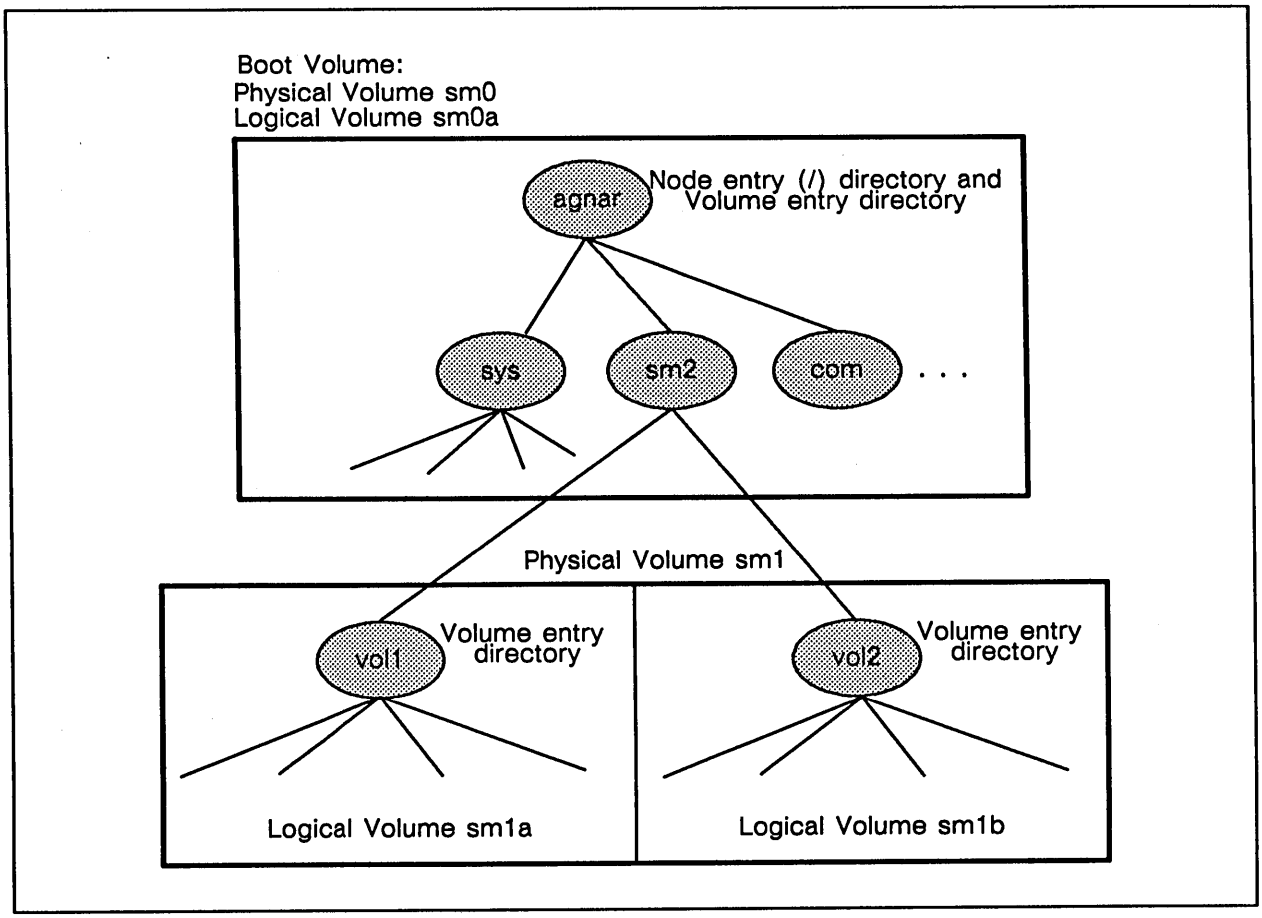


Figure 3-2. A Node with Multiple Physical and Logical Volumes

The 'node_data and / Identifiers

The *DOMAIN System User's Guide* describes directories and links in detail. However, the meaning of *'node_data* (tick-node_data) and / (slash) are especially important in the context of system administration.

The meanings of the identifiers *'node_data* and / (at the beginning of a pathname) are context-dependent. As a result, when used incorrectly (particularly in links), they can result in circular references or references to the wrong files.

The slash character (/), when used at the start of a pathname, always refers to the root 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 ensures that you use the node_data directory that belongs to the node where you are doing your work, and that node only, even if the node is diskless. Similarly, the slash character refers to your node's entry directory, independent of the current working directory.

Some examples of using / and *'node_data* follow

1. *'node_data* and diskless nodes

Suppose there are three nodes: `bigdisk`, `nodisk`, and `diskless`. `bigdisk` is the partner node for `nodisk` and `diskless`. A program or person working at node `bigdisk` who reads the file *'node_data/startup.191* will read `//bigdisk/sys/node_data/startup.191`. A program or person working at node `diskless` who reads the file *'node_data/startup.191* will read `//bigdisk/sys/node_data.e467/startup.191`, where e467 is `diskless`' node ID. A program or person working at node `nodisk` who reads the file *'node_data/startup.191* will read `//bigdisk/sys/node_data.632b/startup.191`, where 632b is `nodisk`'s node ID.

2. **crp**

If you use the **crp** utility to execute commands or programs on another node, the commands execute at the remote node. Therefore, *'node_data* and / refer to the *node_data* and entry directories at the remote node. For example, if you are working at the node `mynode` and use **crp** to create a remote process on the node `yournode`, the following command:

```
$ ld /sys
```

displays the directory listing for *//yournode/sys*, the remote node's */sys* directory.

3. Circular and unexpected references

Because *'node_data* and / have meanings that depend upon the location of the process that makes the reference, it is possible to use pathnames that result in indeterminate or circular references. This is particularly true when you use links to either the / 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 `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, that is:

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

Node Directory Structure

Figure 3-3 shows a typical node directory structure, including system-provided directories 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 required system directories and does not show most links. The following section describes the DOMAIN and DOMAIN/IX system directories.

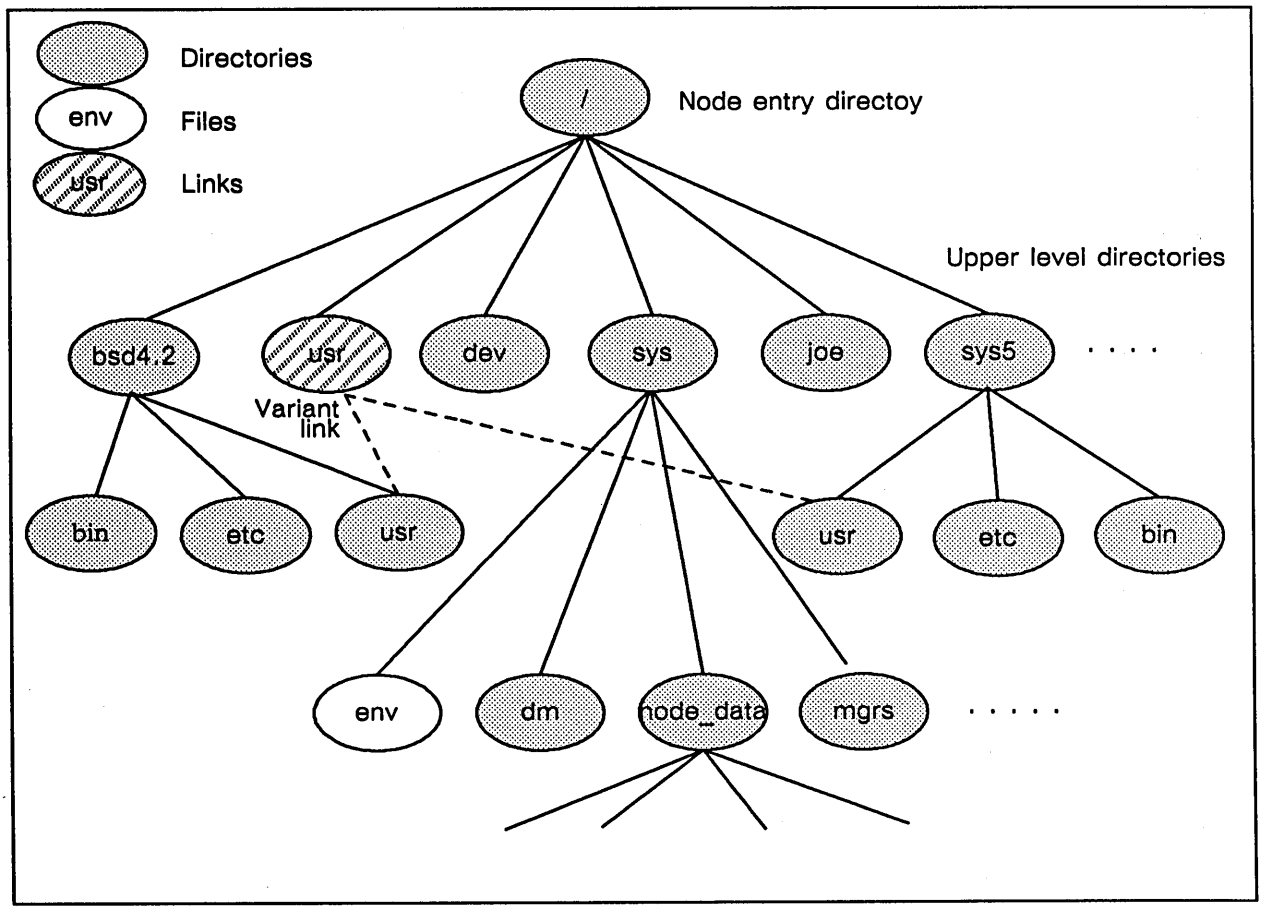


Figure 3-3. Disked Node Directory Structure

DOMAIN System Software Structure

DOMAIN software is organized in the node entry directory, which contains subdirectories and files. The following tables list the directories that are used to create and maintain the network environment. Table 3-1 shows the standard contents of the node entry directory (/) after you have installed system software. If you have optional DOMAIN software, the node entry directory may have additional entries. Table 3-2 shows the directories and files in the system software directory, */sys*, that are used frequently for network management.

Table 3–1. The Node Entry Directory (/)

Entry	Contents
bscom	Bootshell commands
com	AEGIS Shell commands
dev	A link to *'node_data/dev*, the node device file directory
doc	Release notes and update procedures
domain_examples	Example programs for documentation and tutorials
install	Software installation scripts
lib	System libraries
preserve	Files saved during an installation procedure
registry	Created by the **crrgy** (create_registry) procedures
sau[n] *sau1* *sau2* *sau3* *sau4* *sau5* *sau8*	Stand–alone utilities for DN400/420/600 for DN300/320/330 for DSP80/80A/90 for DN460/660/DSP160 for DN550/560/570/580 for DOMAIN 3000 series
sys	System software
sysboot	Node boot program
systest	Online system tests
tmp	A link to *'node_data/tmp*, the node temporary file directory

Table 3-2. The */sys* Directory

Entry	Content
alarm	Alarm server
apollo_logo	Apollo logo file; displayed on node startup
boot	Initial process bootstrap program
cdict	Hashed word list for **fserr**
color [type]_microcode[type] *color2_microcode* *color3_microcode* *color_microcode* *color2_microcode.550*	Microcode for color workstations for DN 580 for DN 570 for DN600, DN660 for DN550, DN560
ctboot	Cartridge tape bootstrap program
dict	Dictionary list for **fserr**
dictdx	Dictionary index for **fserr**
dm	Display Manager files
env	Bootshell environment bootstrap program
gpu1_microcode.a .b	Microcode for DN580 3D graphics accelerator
help	Online HELP files for AEGIS
ins	C, Pascal, and FORTRAN user insert files
mbx	**mbx_helper**, used for inter-process communication
mgrs	Type manager files
net	Network management files
node_data[.node_id]	Node-specific data files
ns	**ns_helper**, used for management of root directories
peb_microcode	Microcode for DN420 floating-point processor
peb2_microcode	Microcode for DN320 floating-point processor
sf	**sf_helper**, used for IPC store and forward
siologin	Serial I/O line servers
source	Source file for various programs
spm	Server Process Manager
subsys	Log-in protected subsystem
sysdev	SIO device descriptor files
traits	System trait files
types	Type definition file

Two subdirectories of the */sys* directory, the Display Manager directory */sys/dm* and the network management directory */sys/net*, are important when creating network services. Tables 3–3 and 3–4 show the contents of these directories.

Table 3–3. The */sys/dm* Display Manager Directory

Directory/File	Contents
color_map	Color mapping information for color nodes
dm	The Display Manager
fonts	Display manager character fonts
input	Standard input file (a null file)
output	Standard output file (a null file)
sbp1	Tablet Server
startup_login[.type]	Node DM Start–up file
startup_templates	Node Start–up file templates
std_keys[n] *std_keys* *std_keys2* *std_keys3*	Standard key definitions for use with AEGIS for 880 keyboard for low–profile Model I keyboard for low–profile Model II keyboard

Table 3–4. The */sys/net* Network Management Directory

File	Contents
diskless_list	List of the diskless nodes that can use this node as a partner
netboot	Diskless node bootstrap program
netmain_srvr	Network maintenance server
netman	Diskless node support server
sample_diskless_list	Example of a *diskless_list* file

The */sys/node_data[.node_id]* Directory

Every node, whether it has a disk or is diskless, requires certain files that are used by that node only. These files include start–up and configuration files such as *inetd.conf* and *startup*[.type]; they also include per–node system files such as backing store for dynamic memory, interprocess communications mailboxes, and device files. The */sys/node_data[.node_id]* directory contains all of these per–node files.

The optional *.node_id* part of the file name enables a disked node to be a partner to one or more diskless nodes. In this case, the name of the diskless node's *node_data* directory includes the node's hexadecimal ID number. For example, if the disked node `ottar` is the partner node for diskless nodes `asmund` (node_id efd3) and `armod` (node_id f2a4), `ottar` would have the following three node-specific directories:

/sys/node_data (for ottar)
/sys/node_data.efd3 (for asmund)
/sys/node_data.f2a4 (for armod)

Table 3-5 shows a listing of a */sys/node_data[.node_id]* directory on a typical DOMAIN/IX node. While it includes most files and directories that you would find in the */sys/node_data* directory, it does not contain entries required by optional or special-purpose software. Similarly, some of these files are not required if you do not implement all DOMAIN/IX features. Many of these files and directories are used only by system software. For information on files and directories that are used by specific application programs or utilities, see the documentation for that software. This documentation includes other chapters in this manual, the *DOMAIN/IX Programmer's Reference for System V*, and *Managing TCP/IP-Based Communications Products*.

NOTE: See "Diskless Node Administration," later in this chapter, for special considerations that are required for the */sys/node_data*.node_id directory for diskless nodes.

Table 3-5. */sys/node_data[.node_id]* **Contents**

File or Directory	Use or User
acl_cache	ACL to permission mapping
alarm_server.msg_mbx	Alarm server
boot_shell	Bootshell
boot_shell_template	Bootshell
crp_mbx001	**crp**
d3m_$error_log	D3M
d3m_$lm_shared_mem	D3M
d3m_$mbx	D3M
data$	System
dev	Device files, a link from /dev
dm_mbx	DM
global_data	System
global_rws	System
hint_file	System
ipc_data	System
mbx_$helper_lock	*/sys/mbx/mbx_helper*
networks	TCP/IP (configuration file)
null_hint_file	System
paste_buffers	DM (contains paste buffer files)

Table 3-5. */sys/node_data* Contents (Cont.)

File or Directory	Use or User
pdb	DM
proc_dir	System (each directory entry is a process name)
proc_dump	System (process crash information file)
ptmp	DPSS/Mail, contains message sent
shell	Bootshell
shell.template	Bootshell
spm_mbx	SPM
stack	System
startup	DM (node start-up file)
startup.1280color	DM (node start-up file)
startup.19l	DM (node start-up file)
startup.color	DM (node start-up file)
startup.spm	DM (node start-up file)
sys_error_log	System, error log
sysmbx	System
tcp_data	TCP/IP
thishost	TCP/IP (configuration file)
tmp	*/tmp* directory
usrtmp	*/usr/tmp* directory
vte_mbx	vt100 emulation
vte_mbx.ctl	vt100 emulation

The DM and Context Inheritance

Whenever you execute a Display Manager command, the DM inherits its context from the last active display window. This context includes all environment variables and the current working directory. This context inheritance has several important effects, including the following:

- The DM does not necessarily inherit the context from the window that currently has the cursor if you have just moved the cursor into the window. You must initiate some activity in the window (if only by pressing the space bar when the cursor is in an input pad) before the DM can recognize the window as "current."

- Any process that you create by executing a DM **cp**, **cpo**, or **cps** command inherits its environment variables from the DM and, therefore, from the current DM context.

- Because the DM context includes the working directory, the meaning of a relative pathname (for example in a **cv** or **ce** command) depends upon the working directory of the most recently active window.

System Resources

You manage system resources by managing files and processes. When you administer a DOMAIN network you normally manage system-wide file resources, such as databases and libraries, and create each user's home directory. Similarly, you manage processes that provide network-wide services (server processes).

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 libraries of application software.

Frequently you do this 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 can be large or heavily used. You should position these resources strategically to maintain an even flow of network traffic and optimize disk space. The placement of servers and the selection of partners for diskless nodes are related issues in system and network management.

You can create an upper-level directory for each user, which can become the user's home directory. A home directory is a 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 create registries as described in Chapter 4. Refer to the *DOMAIN System User's Guide* for a more detailed discussion of naming directories.

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

You create upper-level directories in the same manner as 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

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

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. These processes manage requests from **clients** that are programs or other processes. Clients request access to network resources such as, data, peripheral devices, or communication pathways outside the network.

Server processes often run on **DOMAIN Server Processors (DSPs)**, server nodes that do not have displays and are dedicated to running processes that provide services to other nodes. You can configure network server processes on any kind of node whether or not there are DSPs in your network.

The number of times you implement a server process depends on the particular requirements of your site. You should run server processes that manage network databases, **ns_helper** for example, on several nodes to ensure user access. For example, an **ns_helper** process must run on each DOMAIN network that is connected to make a DOMAIN internet. You might also want to run **ns_helper** on network loops that are frequently separated from a main network.

You also must decide on where to place server nodes within the network topology. Decisions about the placement of server nodes are closely related to decisions about the number of times to implement a server. 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 affects your ability to provide services to nodes and loops as they are switched in and out of the network. Note that a node selected to run a network server process can be used for other activities. It is not necessary, and usually not desirable, to place all network services on one or two nodes. Servers should run on nodes that are stable and secure. (You would not normally run **ns_helper** on a node in an open computing room or a system development node.) Become familiar with the principles of network management and troubleshooting before you determine the numbers and locations of servers. Chapter 6 contains a general description of network servers and the server start-up attributes and options.

Start-Up Files

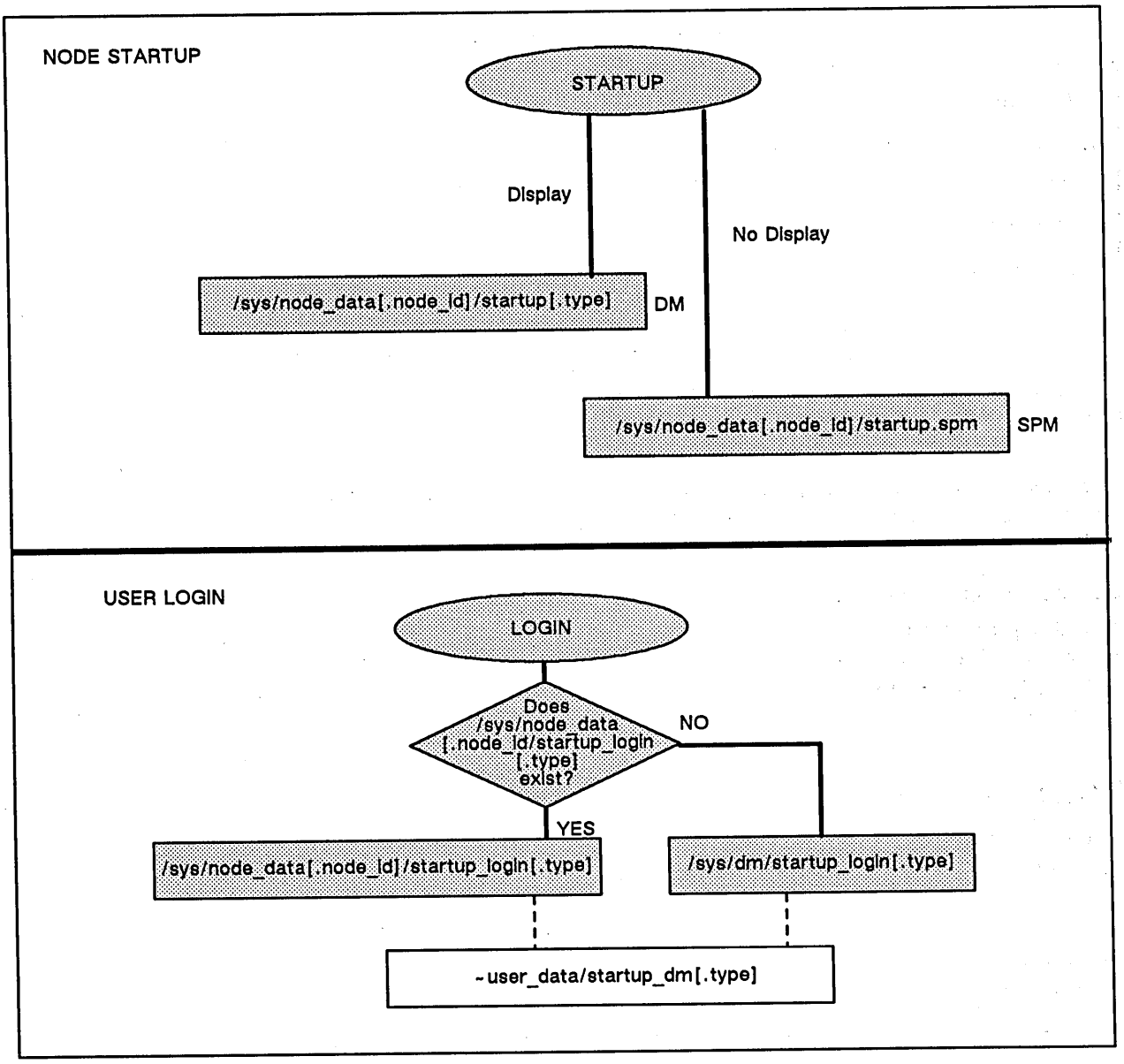
Several types of start-up command files exist: files executed by the operating system at boot time, files executed by the DM and the SPM, and user files executed at log-in time, or when a shell is started.

Table 3-6 lists the command files that can be used at these times. Some of these files run automatically; others must be specified by the automatic start-up files. The following sections describe the files that execute when the DM or SPM start running and when you log in to the DM. The shell documentation (in the *DOMAIN System User's Guide)* describes the shell start-up files and their functions in detail. Figure 3-4 illustrates the relations among the files and the start-up and login procedures.

Table 3-6. Standard Start-Up Files

File	When Run	Comments
/sys/node_data[.node_id]*/startup*[.type]	boot	Runs automatically
/sys/node_data[.node_id]*/startup.spm*	boot	Runs automatically on DSP's.
/sys/node_data[.node_id]/ *startup_login*[.type]	By Display Manager	Runs automatically
/sys/dm/startup_login[.type]	By Display Manager	Runs only if *startup_dm*[.type] is not present
~*/user_data/startup_dm*[.type]	By Display Manager	Must be specified in the node-wide DM start-up file
(User data files are not standard startup files.)		
~*/user_data/sh/startup*	shell startup	

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.



NOTE: Solid lines indicate automatic operations. Dashed lines indicate paths where the preceding file must specify the following file. Program names to the right of the file indicate the program that executes the commands in the file.

Figure 3–4. Start–Up and Log–In Files and Operations

Node Types and Start–Up Files

Because of the differences among display formats, particularly because of differing pixel dimensions, different nodes require different information in their start–up files. For example, the length of the DM windows vary among display types. For this reason, each node and DM log–in 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.19l* is the DM start–up file for nodes with an 800– by 1040–pixel landscape display. Table 3–7 lists the display–type suffixes and indicates the node models to which they apply.

Table 3-7. Start-Up File Suffixes

File Suffix	Node Types
none	DN400, DN420 with portrait display
.1280bw	Monochrome DOMAIN 3000
.1280color	DN580
.19l	DN300, DN320, DN330, DN460, DN550, DN560, DN 570, Color DOMAIN 3000, DN420 with landscape display
.color	DN600, DN660

Note that the installation procedures copy all versions of each start-up file on 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 at nodes with different display types.

Log-out Script Processing

The DM processes log-out scripts. The log-out script must exist in either the */sys/dm* or the *'node_data* directories, and the script must be named *startup_logout.?**, where *?** is the appropriate node type suffix. You cannot start up new processes with the DM **cp**, **cps**, or **cpo** commands from this script.

Template Files

The DOMAIN installation procedures automatically create several start-up template files. Table 3-8 lists these files and describes any special functions they have.

Table 3-8. Start-Up Template Files

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

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; if you delete your start-up file you can copy and edit the template.
- The **netman** process copies them whenever it creates a */sys/node_data*.node_id file for a diskless node. For more details on this see "Administering Diskless Nodes," later in this chapter.

Start-Up File Format

The default versions of the start-up files that are created when you install the DOMAIN software contain most of the commands that you are likely to need. However, most of these commands are

commented out by starting the line with a pound (#) sign. You must delete the pound sign from the start of the line to allow the command to execute. For example, to allow the **netman** process to run whenever the node boots, you would uncomment the following line in the */sys/node_data*[.node_id]*/startup*[.type] file by deleting the pound sign:

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

NOTE: The DM, SPM and the shell are currently case insensitive. However, it is good practice to use lowercase characters when referring to system commands and files.

Any 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 methods described in Chapter 6.

DM/SPM Start-Up Files

The DM/SPM start-up files are run whenever the Display Manager or Server Process Manager processes start executing. The DM normally starts running when you start or reboot a node that has a display. The SPM start-up file executes whenever you start the SPM process on any node.

The /sys/node_data[.node_id]/startup[.type] File

The */sys/node_data*[.node_id]*/startup*[.type] command file, where [.type] is any suffix except ".spm", executes automatically when the DM starts running. You use it to determine the DM operating environment, to specify any server or background processes that must execute on the node, and to run start-up processes that, in turn, use additional start-up files.

You can use these scripts to start any server processes you want to run regardless of log-in and log-out activity. Typically, you start the processes used to provide network services from this script.

Figure 3-5 shows the start-up script we provide for nodes with 19-inch landscape (1024- by 800-pixel) displays. Corresponding scripts for other types of displays draw the locations of the DM windows in different places. Otherwise, the scripts are similar. These start-up scripts are well commented. You should be able to understand the purposes of most command lines from the comments, and later sections in the manual describe the individual server processes in detail. However, the additional information in the following subsection can be helpful in modifying your files.

```
# startup, /sys/dm, default system startup command file for 191, 04/21/83

# Default is black characters on a white (or green) background.
inv -on

(608,774)dr;(1023,799)cv /sys/dm/output
(556,774)dr;(608,799)cv /sys/dm/output;pb
(0,774)dr;(556,799)cv /sys/dm/input

# To enable the diskless node boot server, uncomment the
#              following CPS command.
# cps /sys/net/netman

# To startup default printer
# cps /com/prsvr  -n print_server

# To enable the summagraphic bitpad support, uncomment the
# following CPS command.
# cps /sys/dm/sbp1 /dev/sio2 L

# To startup mbx (IPC) helper
#cps /sys/mbx/mbx_helper

# To properly define the keys for the 880 keyboard,
# uncomment the following command.
#kbd

# To properly define the keys for the low-profile keyboard
# without the numeric keypad, uncomment the following command.
#kbd 2

# To properly define the keys for the low-profile keyboard
# with the numeric keypad, uncomment the following command.
#kbd 3
# D O M A I N / I X  U S E R S :

# Select a default UNIX version for the node by uncommenting
# ONLY ONE of the 'env SYSTYPE' lines:
# env SYSTYPE 'bsd4.2'
# env SYSTYPE 'sys5'

# To use a UNIX style login sequence, uncomment the following;
# env UNIXLOGIN 'true'

# If you DO NOT use UNIXLOGIN, but you still want to be put into
# your project list when you log in, uncomment the following line:
# env PROJLIST 'true'
#
# The file `node_data/etc.rc is a shell script that usually starts
# various UNIX daemons (e.g., rlogind, cron) when the node is booted.
# If you want this script to be run whenever
# your node is booted, uncomment the following line
# cps /etc/run_rc
```

Figure 3-5. 'node_data/startup.191 Script

The /sys/node_data[.node_id]/startup.spm File

The */sys/node_data*[.node_id]*/startup.spm* command file executes whenever the SPM starts running. Therefore, this file can execute in two different circumstances:

- It is the only node start-up file that runs if the node does not have a display (for example on a DSP90).
- It runs in addition to the DM start-up file on nodes with displays that also run the SPM process. Because the SPM allows you to use the **crp** command to create remote processes, nodes such as DN660s that might be used as compute servers often run the SPM.

If the node does not have a display, the *startup.spm* file must do all required node initialization.

If the node has a display, the *startup.spm* file must not include the initialization commands that are already in the *startup*[.type] file. In fact, it is usually best to put all the node start-up commands in the *startup*[.type] file and make sure that there is no *startup.spm* file.

Figure 3-6 shows the first lines of start-up script that we provide in the */sys/node_data* directory when you install DOMAIN software. This script is similar to that in Figure 3-5, but it does not include definitions for creating Display Manager windows. You can edit this script to include any other process that you want to run, regardless of log-in or log-out activity.

```
# startup.spm, /sys/spm,
# default server process manager startup command file
#
# To make sure line 1 listens to xoffs
#
cps /com/tctl -line 1 -insync
#
# To enable the diskless node boot server, uncomment the
# following cps command.
# cps /sys/net/netman
#
# To startup default printer
#
# cps /com/sh -n print_server
#
# To enable the summagraphic bitpad support, uncomment the
# following cps command.
#
# cps /sys/dm/sbp1 /dev/sio2 L
.
# DOMAIN/IX USERS :
.
.
.
```

Figure 3-6. 'node_data/startup.spm Script

System Files Executed at Login

After you log in to the DM at a node, the DM executes one of the following scripts. Note that the SPM does not execute either script during remote logins.

- */sys/node_data*[.node_id]*/startup_login*[.type] (where type is the display type)
- */sys/dm/startup_login*[.type], only if there is no */sys/node_data*[.node_id]*/startup_login*[.type] file for the node

NOTES: The SPM does not execute either of these scripts on remote login.

See the discussion of **siologin** in Chapter 6 for information on start-up files executed when you log in over an SIO line.

The *startup_login*[.type] script should start the processes that need to be invoked every time someone logs in. Examples include processes that draw a window and create a shell or run a user-specific login script. Processes specified in the *startup_login*[.type] file stop running when the user logs off.

If you want a specific node to have a specific *startup_login* file, put a *startup_login* file in the */sys/node_data*[.node_id] directory. For example, if you want diskless node `e37d` to have a special *startup_login* file, create a file named `/sys/node_data.e37d/startup_login.191`. In this case, the operating system runs the node-specific login file and does not run the file in */sys/dm* whenever someone logs on to node `e37d`.

If you do not put a *startup_login*[.type] file in the */sys/node_data*[.node_id] directory, the operating system executes */sys/dm/startup_login*[.type], the default script supplied with your node. Therefore, it is unnecessary to create *startup_login* files for each node ID if a standardized file (for each display type) is sufficient. In any case, it is good practice to keep a */sys/dm/startup_login*[.type] file for each type, both as a template file and to support any diskless nodes that might use the disked node as a temporary partner.

Figure 3-7 shows the */sys/dm/startup_login.191* script that we provide. This script creates an AEGIS shell process by default. You can use the default shell window, comment it out by adding a pound sign (#), change it to draw the shell windows in a different location, or replace it with a command to start a Bourne or C shell. Uncomment the last line in the file to run the per-user login command file.

```
# startup_login (the per_login startup file in `node_data or /sys/dm)

# main shell whose shape is generally agreeable to users of this node
(0,500)dr;(799,955)cp /com/sh

# and the users private dm command file from his home directory's user_data
# sub-directory. Personal key_defs file is also kept in user_data by DM.
# cmdf user_data/startup_dm.191
```

Figure 3-7. /sys/dm/startup_login.191 Script

User Files Executed at Login

If you remove the comment character (#) from the last line in the */sys/dm/startup_login* script, the DM looks for, and executes, *~/user_data/startup_dm*[.type] (where ~ is the user's naming directory). If you remove the pound sign from the *startup_login* script but do not create a */user_data/startup_dm*[.type] file, an error message appears at login. The SPM does not execute this script during remote logins. We do not supply a *startup_dm* file, but you can create the script by using DM commands. The *DOMAIN System User's Guide* provides more information about this script.

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 sections describe the rules and techniques for managing diskless nodes and their partners. The last section in this chapter describes a procedure for configuring a diskless node's partner.

Diskless Node Operation

When a diskless node displays the log-in prompt, all the programs required for its operation are in place. While the *DOMAIN System User's Guide* 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 "request for volunteer" broadcasts. It checks its */sys/net/diskless_list* file for the requester's hexadecimal node ID. If the ID appears in the file, it answers the diskless node's request.

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_name/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 Display Manager executes the commands in the *//partner/sys/node_data.diskless_node_id/startup*[.type] file.

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

Establishing Diskless Nodes and Partners

A diskless node's partner node provides the system software and disk services for the diskless node. It 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 user nodes with display monitors, or DSPs, the server nodes.

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 */sau2* directory and a */sau5* directory. Similarly, the partner's */sys* directory must have any microcode files required by the diskless node. Tables 3–1 through 3–3 list many of the node-type dependent system software files.

Each diskless node has its own */sys/node_data.diskless_node_id* directory on the partner node, and *'node_data* 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

You control the assignment of diskless nodes to partners through the */sys/net/diskless_list* file, such as the one shown in Figure 3-8. A disked node can be a partner to several diskless nodes. The disked node will volunteer to be a partner for any diskless node whose node ID is in the disked node's *diskless_list*. 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.

```
# 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 3-8. A /sys/net/diskless_list File

NOTE: If you put a diskless node 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 therefore its contents, could change whenever the node reboots. Therefore, you must configure the diskless node correctly on *each* possible partner node. (See the following section "The */sys/node_data.node_id* Directory on New Partners" for more details.)

If you use names for diskless nodes, do not change the name of the */sys/node_data.diskless_node_id* directory to */sys/node_data.diskless_node_name*. Remember, a diskless node name is not valid in a pathname. You must specify the diskless node ID to accurately access this object.

When you boot a diskless node you can request a specific partner node. See "Requesting a Specific Partner," later in this chapter.

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.

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. Note that if a service representative installs a diskless node, he or she creates a partner for the node; however, you might want to use a different partner.

PROCEDURE 3–1: Providing a Permanent Partner for a Diskless Node

1. Determine the diskless node's ID number. 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.
   ```

2. Log in to the partner node. If the partner does not have a display you can create a remote shell on the partner by using the following command:

   ```
   $ crp -on //partner -me
   ```

3. Add the diskless node's node ID to the partner node's */sys/net/diskless_list* file.

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

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

   a. Remove the comment character (#) from the following line in the partner's */sys/node_data*[.node_id]*/startup*[.type] file:

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

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

   b. 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 in its DM input window:

      ```
      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 Step 2.)

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

6. You can limit the size of the partner's memory pool that is available to the diskless node for paging requests. Enter

   ```
   $ netsvc -p [pool_size]
   ```

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

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

Procedure 3-1 (Cont.)

- If you do not use **ns_helper** on your network enter the following command. After you finish this procedure, catalog the node on the network as described in Chapter 2.

  ```
  $ ctnode node_name node_id
  ```

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

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

8. Create the diskless node's node start-up and configuration files. While there are many different ways you can do this, the following steps will work in all cases:

   a. Create the diskless node's */sys/node_data*.node_id directory by entering the following command at the partner node:

      ```
      $ crd /sys/node_data.node_id
      ```

      where node_id is the diskless node's ID.

   NOTE: If you are changing the partner of an existing diskless node, you can skip Steps 8b and 8c. Instead, copy these configuration files from the old partner to the new partner.

   b. Copy the*/sys/node_data*.node_id/*startup*[.type] file (where type indicates the type of display on the diskless node, not the partner) from the partner. You can copy the file from the partner's */sys/node_data* directory, or from the */sys/dm/startup_templates* directory (for diskless nodes with displays) or */sys/spm/startup_templates* directory (for diskless DSPs).

   c. Edit the following files to meet the diskless node's needs:

      /sys/node_data.node_id/startup[.type]

9. Log off the partner node.

10. You can now start or restart the diskless node. If you are changing an existing diskless node's partner, do the following:

    a. Enter the following DM command to log off and shut down the operating system.

       ```
       Command: shut
       ```

    b. The Bootstrap PROM prompt (>) appears on the screen; enter the following reset command:

       ```
       > re
       ```

    c. Press <RETURN> twice.

    d. The PROM identifier appears, followed by the PROM prompt. Restart the operating system by entering the following command:

       ```
       MD REV n, yy/dd/mm
       > ex aegis
       ```

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

END OF PROCEDURE 3-1

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. **Netmain_srvr**, the network maintenance server, and the **netmain** interactive tool, along with the **netsvc** shell command, can help you to manage partner assignments.

Diskless Node Management Commands

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. See *Managing DOMAIN Networks* for a detailed description of **netmain_srvr** and **netmain**.

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 reset the screen on the diskless node and eliminate the messages. You must reboot the diskless node whenever its partner node reboots.

Warning of a Partner Shutdown

It is good practice for the administrator of a partner node to notify diskless node users of intended shutdowns. Use the **send_alarm** command with the **-di -mydi**, or **-din** option to notify users of shutdowns. For example, if your are shutting down your node, `disked`, the following command will warn all diskless nodes that use your nodes 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 another disked node that runs **netman** as its partner, even if that node does not have the diskless node on its partner list. You can also use this procedure to boot a node that has a disk as a diskless node, for example if the node's system software is corrupted. 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<RETURN>
   ```

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

3. Enter the following command to restart the node:

   ```
   > ex aegis<RETURN>
   ```

Chapter 4

Creating and Maintaining User Accounts

The registry is a distributed database that identifies legitimate users of the network. The registry allows you to

- Protect the system from unauthorized use by persons without valid accounts
- Provide users with home directories and automatically start processes for users when they log in

A network registry is not required, but networks without registries are open, and you cannot control user access in an open network. Because of this, we assume that most system administrators will choose to create a registry in order to maintain a secure network. If your site is a multiple ring environment, there are considerations for managing registries that are beyond the scope of this book. Refer to *Managing DOMAIN Internets* for information on registries in multiple ring environments.

In addition to providing security for your network through registries, you can protect individual nodes in the network with procedures described in Chapter 5. This chapter describes

- How the registry controls the user's access to the network
- The parts of the registry database
- How to create and maintain a network registry at your site

System Operation with the Registry at Login

When a user logs in, the operating system looks at the files that form the registry database. These files are called the **person, project,** and **organization** files (**ppo**), and the **account** file (**acct**). You can think of the ppo files as a "master menu" that contains all the responses to a log-in prompt that are possible at your site. For example, the person file contains all the log-in names at your site; the project file can contain the names of projects or products, for example, "System_backup" or "Widget_design." The organization file can contain the names of groups, e.g., "Laboratory," "Marketing," or "Accounting." An optional **full_names** file, to list the full names of users associated with their log-in account names, is also part of the registry database.

The account file contains those selections from the ppo files that make up a user's log-in account(s). For example, user John Jones might have several accounts under the same or different log-in names. Each of his accounts can specify a different home directory and can have a different start-up script. In another case, any number of users (log-in accounts) can share the same project name. Those users might want their working directory set to the directory of their project at login. You arrange these options for users through the registry account file.

The account file also contains the user's password. You can put passwords in user accounts, or users can create their passwords and change them from session to session. If the user changes a password, the operating system updates the account file.

If the account given to the log-in prompt is not in the account file, the system refuses to allow the user to login. However, the account file contains the default log-in name "user.none.none". Any person can log in to the network as "user.none.none", or simply "user". With the user identification provided by the registry, however, you can control access to files and directories on the network.

Since it performs these important "gatekeeping" functions, the registry database or site directory (*/registry/rgy_site*) as it is called, must be available at all times. It is not practical to rely on a single node, containing one site directory, to be in the network at all times. Therefore, the registry creation procedure distributes site directories to the nodes that you specify in the master registry file.

The master registry file (*/registry/rgy_master*) is a master list of the pathnames of site directories. The shell command **lrgy** displays the master registry file once you have created the registry. Figure 4-1 shows an example of the master registry file.

```
 $ lrgy<RETURN>
Registry:
//rose/registry/rgy_master Sites of registration data files:

        //rose/registry/rgy_site1
        //tulip/registry/rgy_site2
        //lilac/registry/rgy_site3
```

Figure 4-1. Example of a Master Registry File

To locate the registry sites, every node in the network has its own copy of the master registry file. The node's master registry file copy is called */registry/registry*.

When a user tries to log in, the system reads the node's registry file copy to obtain the pathname of a registry site directory. The system looks for the first available registry site directory on its list. Next, it checks the site directory's account file to verify the user's identity. When a user enters a valid log-in name and associated password, the system does the following:

- Sets the working directory to the home directory
- Sets the naming directory to the home directory
- Executes the user's start-up files if they exist
- Updates the node's local registry

Figure 4–2 shows the actions of the operating system as a user logs in.

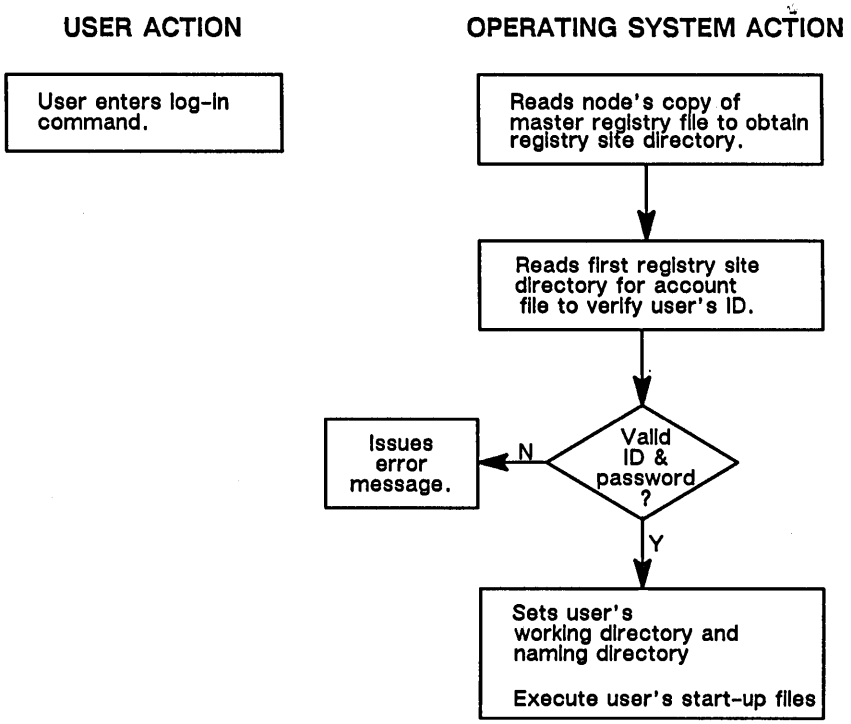


Figure 4–2. Operating System Action During Normal Login

The Local Registry

If a user logs in when the network registry is unavailable (i.e., the system tries to find the registry sites listed in */registry/registry* but cannot reach any of them), it uses the local registry as a last resort.

The local registry is a system-maintained database that describes the node's last set of users and records the date and time when they last used the system. You create the local registry whenever you bring a node into the network. Each local registry resides in the node's */registry* directory.

The local registry has one registry site directory (*/registry/local_site*)and one master registry (*/registry/local_registry*) file. A local registry resides on every node in the network. The account file in */registry/local_site* stores the log-in accounts of the node's last users.

The system maintains the local registry database. Every time someone logs in using the network registry, (i.e., "normal operation"), the system updates */registry/local_site*, on that node, with current information from the network registry database.

When the system uses the local registry, it reads */registry/local_registry* to get the pathname */registry/local_site*. It then checks the account file in the */registry/local_site* directory. If the user has logged in to the node recently with the account given to the log-in prompt, the system finds the information in the account file and logs the user in. If the account file has no information on this user's account, the system refuses to log the user in.

If both the system registry and the local registry are unavailable, the system will automatically log the user in as "user.none.none," the system-wide default identity. This mechanism guarantees that you can log in at all times.

Figure 4-3 illustrates how the operating system uses the network and local registries to grant access to the system.

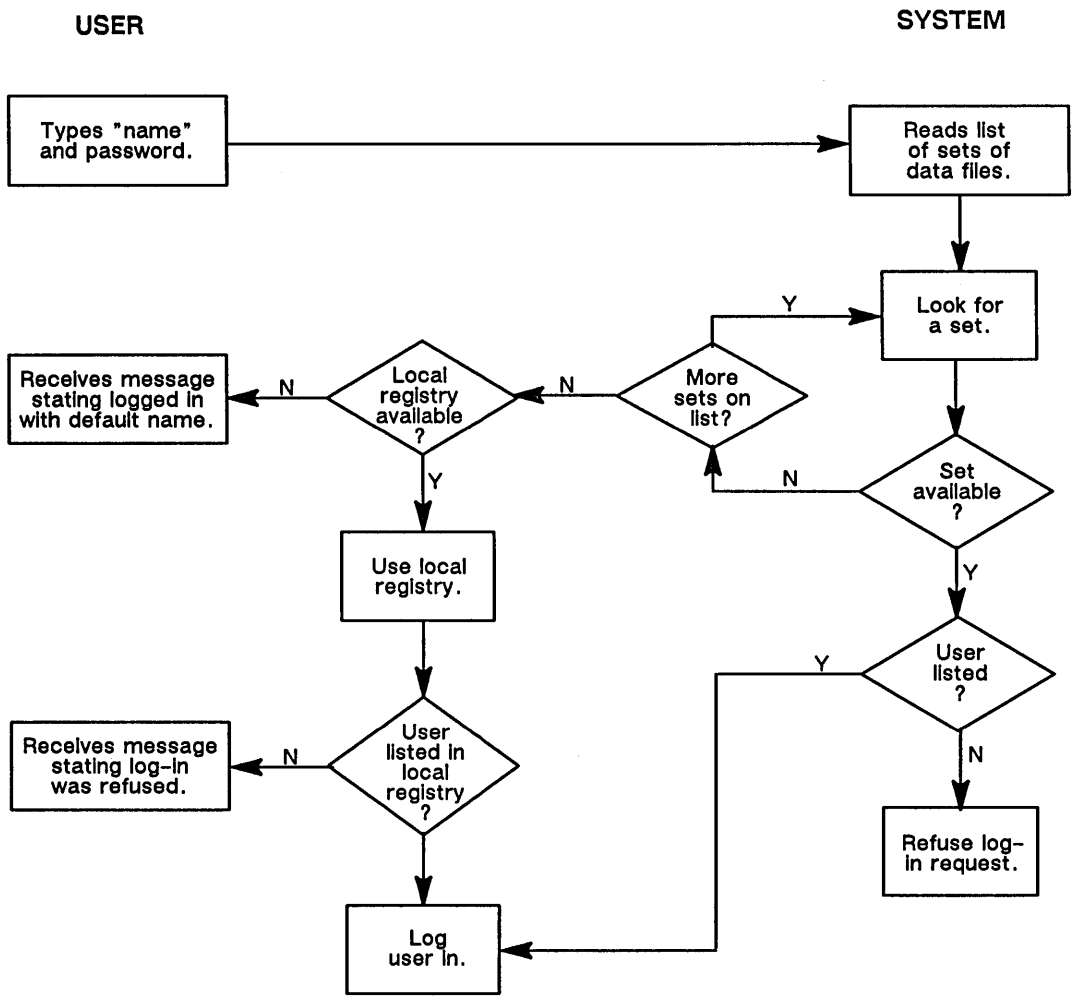


Figure 4–3. Using the Local Registry to Grant Access

Considerations for Implementing Your Network Registry

As system administrator, you must make several decisions about the network and local registries, and you must ensure that access to your registry is properly controlled. Before you create the registry, decide on

- The number of registry site directories
- Registry site directory locations
- Registry site names
- Registry master file location
- Number of user accounts stored in local registries
- Time period of valid accounts in local registries

Most installations will find two or three site directories sufficient to protect access to registries in case of node failure. If you have a multiloop network, place the registry site directories on different loops to maintain registry service as loops are switched out of the network. Users on loops with registry site directories always have access to a site. Users on other loops will have access to at least one other site. The local registry can maintain service for most users when loops without site directories are switched out of the network.

Select stable nodes that always maintain their network services (**netsvc -a**) as registry site nodes. To minimize the possibility that a node will not maintain its network services, ensure that users of these nodes do not use the **netsvc** command with **-n** or **-l** options. These options prevent the node from responding to network requests that originate at other nodes. You can use ACLs (Access Control Lists) to prevent any user from executing the **netsvc** command at the local node. A later section of this chapter describes how to set ACLs on the local copy of **netsvc**.

When you edit the registry site data files, you can do so successfully only when all site directories are available. Therefore, a loop or node that is often switched out of the network is not a good location for a registry site directory.

If you change the location of a site directory, update */registry/rgy_master*, the master list of pathnames. Also update each */registry/registry* in the network (the node copy of the registry master file). If a node (or loop containing the node) is off the network at the time of the update, its copy of */registry/registry* won't get updated. The problem this situation can create may not become apparent until long after you've done the update.

As described above, during a login, the system looks in the list of pathnames for the first site directory, then looks for the next pathname if it doesn't find the first directory. Assume that some node's */registry/registry* file contains information that is only partially current. During network difficulties, the operating system might not reach any of the current site directories. Moreover, because */node_name/registry/registry* contains outdated information, the operating system has fewer nodes to search for current site directories. A user might have difficulty logging in.

If the same node fails to receive new information over the course of several updates, its */registry/registry* file can become so outdated that some user won't be able to log in. To prevent this, follow proper registry maintenance procedures, which include updating */registry/rgy_master* and every */registry/registry* each time you change site directory locations. Of course you should try to prevent these occurrences by selecting good locations for registry sites and keeping the registries at those locations.

Although you cannot add names to the local registry, you can use the shell command **crrgy -loc [n] -days [n]** to specify the number of accounts to be stored in the local registry and the time period that entries in the local registry will remain valid. You can use the shell command **lrgy** to display the local registry specifications on any node. Figure 4-4 shows the output of this command.

```
$ lrgy -loc<RETURN>
Registry: //rose/registry/local_registry
Sites of registration data files:
        //rose/registry/local_site
Registry is LOCAL. It has 25 slots for login;
the expiration period is 21 days.
```

Figure 4–4. The Local Registry

You can also replace (delete and re-create) the local registry. Note that you lose the history of the node's last users if you replace the local registry. A new history will begin building immediately. Use **edacct –l –loc** to examine the local registry on any node. Figure 4–5 shows the output of this command.

```
$ edacct -l -loc<RETURN>
martin    none       eng      52D1 85/04/22.08:48exp:85/05/14 //jay/martin
marks     dev        lab      52D1 85/04/11.19:10exp:85/05/03 //robin/marks
user      none       none     52D1 85/04/19.10:39exp:85/05/11 /
brown     sys_admin  eng      52D1 85/04/01.17:36exp:85/04/23 //duck/brown
```

Figure 4–5. The Local Registry Account File

Because the system only performs updates to */registry/local_site* on the current node, the following scenario is possible.

1. Assume node A does not contain a network registry site directory. A user logs in at node A with password "X."
2. The same user logs in at node B and changes the password to "Z."
3. The user logs in at node A when the network registry is unavailable and the system must use the local registry. The node A local registry requires the password "X," not "Z," because the system does not update node A's local registry when the user logs in to node B, and the system can't find out about password "Z" since the network registries are not available.

However, if the user changes a password, then logs in to node A at a time when the network registry is available, the system will update node A's local registry, and the account record will contain the password "Z."

Network Registry Objects

The following section describes the structure of the network registry. The network registry is a distributed, replicated database that allows a user to access the network under normal conditions. The network registry has the following structure:

- Master registry file (one per network).
- Site directories (one or more per network).
- Registry file copy (one per node). Table 4–1 lists the locations and standard names for the network registry files and directories.

Table 4-1. Registry Object Standard Names and Distribution

Object	Standard Name	Data Location and Number
Master Registry File	*//node_name/registry/rgy_master*	Only one per network
Site Directory	*//node_name/registry/rgy_site[n]*	At least one per network
Registry File Copy	*//node_name/registry/registry*	One on each node in network

We strongly recommend that you use standard names for these objects even though */registry/registry* is the only required name for correct operation of the registry. Append a name or number to */rgy_site* to differentiate site directories. Figure 4-6 illustrates the distribution of registry objects in a small network.

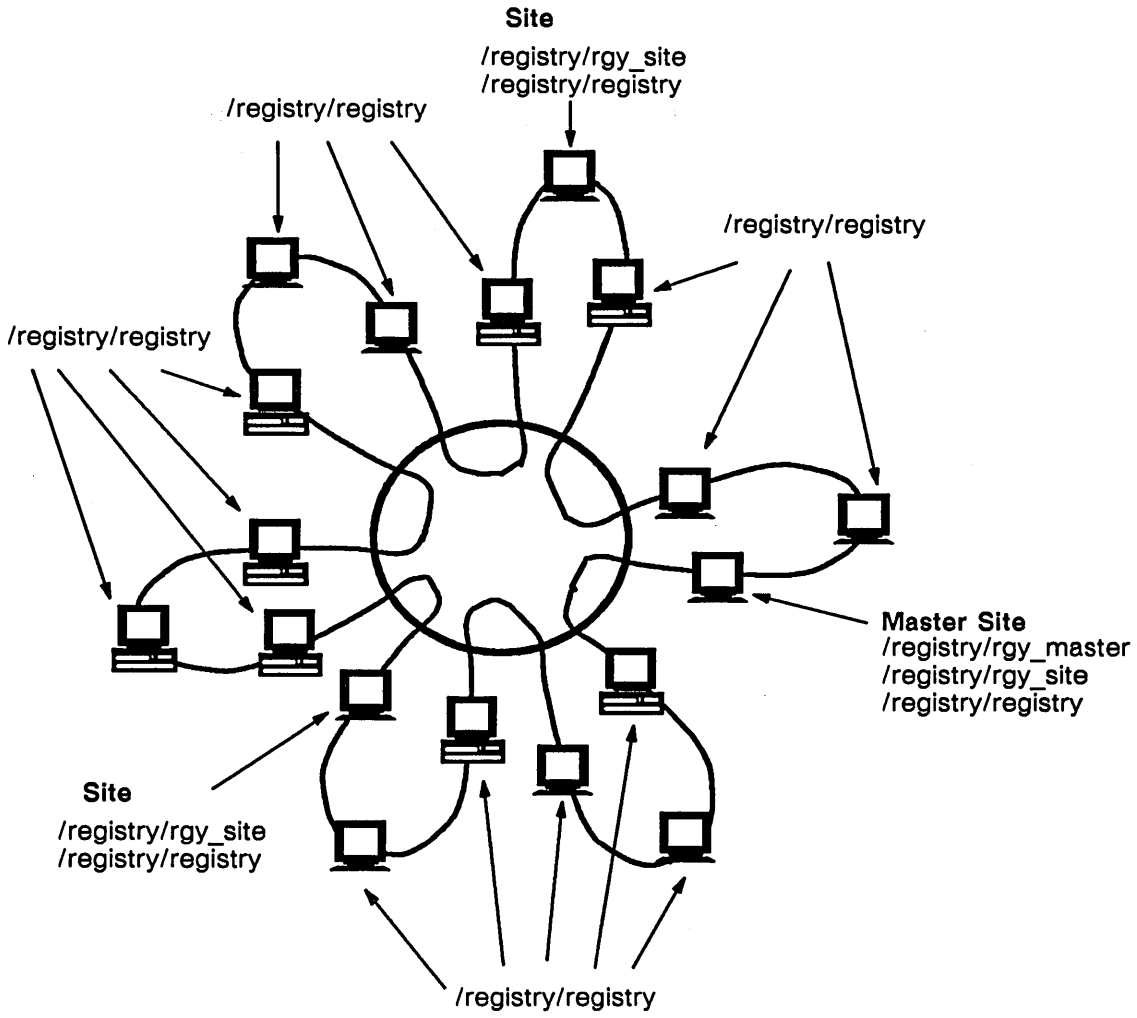


Figure 4-6. Distribution of Registry Objects in a Network

The Site Directory and Associated Data Files

Each **site directory** contains a copy of the database, the files that list authorized network users. Each node that has a site directory is a **site node** or simply a **site**. All site directories are identical. Each */registry/rgy_site[n]* directory contains five data files: **person, project, org, account,** and **full_names**. These data files identify the people, projects, and organizations that may use the network. Figure 4–7 illustrates the contents of these files.

/registry/rgy_site

person	project	org	full_names
name1name-id1	proj1proj-id1	org1org-id1	full name 1name2
name-id2	proj2proj-id2	org2org-id2	full name 2
:	:	:	:

account

name-id1	proj-id1	org-id1	password /home_directory
name-id2	proj-id2	org-id2	password /home_directory
name-id2	proj-id1	org-id2	password /home_directory
name-id3	proj-id1	org-id1	password /home_directory

Figure 4–7. Registry Data Files

The person, project, and org files, collectively known as the ppo files, contain 1– to 32–character string names and corresponding, unique, identification numbers. The system builds user accounts from these representations. Because each name has its own number, the same string name could represent a person, project, and an organization. For example, you could have person: Jones; project: Jones; and org: Jones. However, we recommend that you use different names for clarity. The registry also provides a full_names file that (optionally) associates a 32–character string with any name in the person, project, or org files. Typically, this file is used to associate a user's full name with the ppo names.

The registry account file contains records that associate one entry each from the ppo files with a password and home directory. The order of records in the account file is significant. The system uses the first record it finds that matches the information given by the user to the log–in prompt. A user may have accounts for two projects and want to log in to one of them by default. Manipulate the order of entries in the account file to achieve the result desired.

For example, user John Jones has project accounts "widgets" and "washers," and he wants to log in to his "widgets" account by default. Manipulate the order of entries in the **account** file as follows:

```
jones    widgets   none    password        //sam_node/jones/widgets
jones    washers   none    password        //sam_node/jones/washers
```

Now, if Jones logs in as "jones," he gets the first matching identity, jones.widgets.none. He logs in as "jones.washers" to get his second account.

Enter names in the person, project, and org files with the **edppo** shell command. Enter accounts in the account file with the **edacct** shell command. (Figure 4–10 shows examples of command usage.)

There are default names in the ppo files that are useful in system management. Their specific purpose is discussed below. Table 4–2 shows the default names in the person, project, and org files.

Table 4-2. Default ppo Names

person	project	organization
root	backup	apollo
sys_person	locksmith	none
user	login	sys_org
	none	
	server	
	sys_admin	
	sys_proj	

There are also default accounts which come with the system, and these are shown in Table 4-3.

Table 4-3. Default account Names

person	project	organization	home directory
user	none	none	/
user	sys_admin	none	/
user	backup	none	/
user	none	apollo	/
user	locksmith	none	/
user	server	none	/

The default ppo names and accounts are expected by certain system operations. For example, server processes created with the cps command receive the name "user.none.none." Chapter 5 describes how this account allows all system processes to use the server. The project name *login* belongs to the "login" protected subsystem, described in Chapter 5. The person name *root* belongs to DOMAIN/IX.

The ppo names *sys_person*, *sys_project* and *sys_org* can be used as required by the needs of your organization. The organization name *apollo* is for service representatives who log in to your system.

The **project** names *sys_admin*, *backup* and *locksmith* are for system administration purposes. The *sys_admin* account is general, the *backup* account is specifically for backing up system and users files, and the *locksmith* account can override all ACL rights assigned to any file or process. The account "user.none.none" is the system default log-in account.

Note that passwords for these accounts are not supplied. To further limit system access, assign passwords to all of the default system administration accounts. The user.none.none account is discussed in Chapter 5.

The Master Registry File

One node in the network contains the master registry file. See "System Operation with the Registry at Login" earlier in this chapter for a description of the master registry file. Any node may contain this file, but it is convenient and recommended to place it on the node that contains one of the registry site directories. This node is called the **master site node.**

Each Node's Registry File Copy

The node's registry file copy is a copy of the master registry file. Every node in the network, including the node holding the master registry file, must contain a registry file copy.

Each Node's */registry* Directory

Each node's */registry* directory contains the registry file copy, a local registry file and local registry site directory. The shell command **ld** lists the contents of this directory, as shown in Figure 4–8.

```
$ ld /registry<RETURN>
Directory "/registry":
local_registry     local_site          registry
3 entries.
```

Figure 4–8. A Node's /registry Directory

On site nodes, the */registry* directory contains the subdirectory */rgy_site[n]*. The contents of */rgy_site[n]* are displayed in Figure 4–9.

```
$ ld rgy_siten<RETURN>
Directory "rgy_site"
account   full_names          org       person    project
5 entries
```

Figure 4–9. Example of a rgy_site Subdirectory

Creating and Maintaining Registries

This section reviews the commands used to create and maintain registries and gives general information about their use. It also provides procedures used to create and maintain registries.

Shell Commands for Managing Registries

Table 4–4 lists the commands that create, update, and maintain the registry database.

Although all registry shell commands take a **–r** option that operates on a specific registry pathname, we strongly recommend that you not use this option. Instead, allow these commands to use the default, i.e., to use */registry/registry* to locate the master file and site directories. The one exception to this recommendation is for using the **lrgy** command to examine the contents of the */registry/registry* file on a given node. In this case, invoke

$ lrgy –r //node_name<RETURN>

or, to examine the local registry on a given node, enter

$ lrgy –r //node_name –loc <RETURN>

Table 4-4. Registry Administration Commands

Mnemonic	Command	Purpose
CREATE REGISTRY	**crrgy**	Creates a registry that includes a master file of site pathnames, any number of site directories and account files. These files contain default names and accounts. Use the command to create local registries or to change registry sites. You must have write access to the registry database and its directories to use this command.
EDIT PPO	**edppo**	Creates, edits, and lists the names and associated full name text strings in the **ppo** files. Use **edppo** as a one-line command or as an interactive editor. You must have write access to the registry database to edit **ppo** files.
EDIT ACCOUNT	**edacct**	Defines, changes, and lists accounts. Use it as a one-line command or as an interactive editor. You must have write access to the registry database to edit accounts.
LIST REGISTRY	**lrgy**	Lists the contents of a registry file, including the pathname of the registry master file and the pathnames of all registry sites.
SALVAGE REGISTRY	**salrgy**	Salvages a registry. Use **salrgy** to ensure that all sites of a registry contain the same information You must have write access to the registry database to use this command.

Creating Site Directory and Master Registry Files

When you create a registry for your network, execute Procedure 4-1 at the node chosen to contain the registry master file (*/registry/rgy_master*), and a registry site directory (*/registry/rgy_site[n]*). This node is called the **master site node**. Procedure 4-1 uses a shell script to create the following objects at site nodes:

- */registry* directory
- Master registry file, called *rgy_master* and registry site directory, called *rgy_site[n]*
- A copy of the registry master file, called */registry/registry*
- A local registry

After you create the *rgy_site* directories, use **edppo** and **edacct** in their interactive modes to enter names in the person, project, organization, full names, and account files. As a last step, use another shell script to replicate the registry on the other nodes selected as sites. Procedure 4-2 creates the necessary registry files and directories on all the nonsite nodes in the network.

Protect your network registry files and directories from unauthorized access as soon as you create the registry with Procedures 4-1 and 4-2. Before executing Procedures 4-1 and 4-2, read the information in Chapter 5 on protecting system software and registries. Then execute the protection procedures immediately after creating the registries.

PROCEDURE 4-1: Creating the Registry Database on the Site Nodes

NOTE: The */install* directory must be available to complete this procedure.

Execute this procedure on the master site node. Ensure that all the nodes are cataloged before executing this procedure.

1. At the master site node, run the **init_master_rgy** shell script in the installation utility directory */install* by typing

   $ /install/init_master_rgy<RETURN>

2. Create */registry* directories at other site nodes and add the site pathnames to the registry. For *//site_node_name_1,2...*, substitute the names of the nodes that you have chosen as site nodes. Type the following:

   $ crd //site_node_name_1/registry<RETURN>
   $ crd //site_node_name_2/registry<RETURN>
   $ crd //site_node_name_3/registry<RETURN>
   $ crrgy -a //site_node_name_1/registry/rgy_site1<RETURN>
   $ crrgy -a //site_node_name_2/registry/rgy_site2<RETURN>
   $ crrgy -a //site_node_name_3/registry/rgy_site3<RETURN>

3. Use **edppo** to put names in the ppo files. **Edppo** accepts 1- to 32-character string names, and 32-character strings for associated full-name text. To add a name and optional full-name text, type

   $ edppo -a name [fullname]<RETURN>

   To add a project, type

   $ edppo -proj -a project_name<RETURN>

   To add an organization, type

   $ edppo -org -a organization_name<RETURN>

   The example at the end of this section shows **edppo**'s interactive mode. It is more suitable than the command line when adding multiple names.

4. Create user accounts and assign passwords and home directories with the **edacct** command. Type the following:

   $ edacct -a pers proj org homedir password<RETURN>

   In interactive mode, **edacct** prompts for each account's **ppo** names, assigned password, and home directory. Figure 4-10 shows **edacct**'s interactive mode. It is more suitable than the command line when entering multiple accounts.

5. Create a registry file copy and local registry for the site nodes, using the **init_rgy** shell script in the installation utility directory */install*. Specify the master site node's entry directory name, and the site node names, with the double slash (*//*):

   $ /install/init_rgy //master_site_node_name //site_node_name<RETURN>

6. Read the information on protecting registries in Chapter 5. Then use Procedure 5-1 to protect the registry database you have created on each of these site nodes.

END OF PROCEDURE 4-1

PROCEDURE 4-2: Creating Node and Local Directories

Create the remainder of the registry database on nodes that are not site nodes. Use this procedure at every nonsite node in the network. Do the procedure after Procedure 4-1.

1. At every nonsite node in the network, create a registry directory, registry file copy, and local registry with the **init_rgy** shell script in the installation utility directory, */install*. Specify the master site node's entry directory name, with the double slash (//) before it, as follows:

   $ /install/init_rgy //master_site_node_name<RETURN>

2. Go to Chapter 5 and use Procedure 5-1 to protect the registry files and directories on each node.

END OF PROCEDURE 4-2

Sample Session for Creating a Registry

Figure 4-10 shows a sample session, using the interactive forms of the **edppo** and **edacct** shell commands to define ppo names and create user accounts in a new registry.

```
$ edppo => l                                          # List the names in
                                                      # rgy_site

  root  sys_person   user
=> a                                                  # Add new names
add=> jones 'John J. Jones'
add=> donnell 'Rose Donnell'
add=> sas 'Susan Smith'
add=> jih
Enter full name in quotes: 'James I. Harris'          # prompts for full name
add=> pjk
Enter full name in quotes:<RETURN>                    # full name is optional
add=><RETURN>                                         # to end adding names
=> l                                                  # List the names in
                                                      # rgy_site
jih    jones        donnell     pjk    root
sas    sys_person   user
=> lf                                                 # list with full names
jih               James I. Harris
jones             John J. Jones
donnell           Rose Donnell
pjk
root
sas               Susan Smith
sys_person
user
```

Figure 4-10. Sample Session

```
#  Up to this point no information has been written to rgy_site.
#  The next step is critical because we actually write out the changes using the
#  "wr" interactive command.

=> wr                                                    # update the file and
                                                         # quit
$ edppo -proj                                            # define two projects
=> a finance 'finance'
=> a mkte 'Marketing, East Coast'
=> wr                                                    # update the file and
                                                         # quit
$ edppo -org -a ajax_distrib                             # define one org as a one
                                                         # line command


$ edacct                                                 # define some accounts
                                                         # using the ppo names
=> la                                                    # list all entries
user      none       none      /
user      sys_admin  none      /
user      backup     none      /
user      none       apollo    /
user      locksmith  none      /
user      server     none      /
=> t                                                     # go to top of file
=> a jih finance none //fin/jih ' '                      # no password
=> a jones mkte none //mkt/jones 'john'                  # assigned password
=> a donnell  none none //od/rose ' '
=> a donnell  sys_admin none /  ' '
=> ln                                                    # list next entry
user           none           none          /
=> lc                                                    # list new or changed
                                                         # entries
jih            finance        none          //fin/jih
jones          mkte           none          //mkt/jones
donnell        none           none          //od/rose
=> wr                                                    # update the file and
                                                         # quit
# we now have the following accounts in this registry:
$ edacct -l
jih            finance        none          //fin/jih
jones          mkte           none          //mkt/jones
donnell        none           none          //od/rose
user           none           none          /
user           sys_admin      none          /
user           backup         none          /
user           none           apollo        /
user           locksmith      none          /
user           server         none          /
```

Figure 4–10. Sample Session (Cont.)

Maintaining Registries in Existing Networks

You are responsible for maintaining and protecting your registry database. This section provides guidelines for registry maintenance and protection. See Table 4–4 for a summary of the commands used to update the user account database and to add and delete registry sites. See Chapter 5 for more information about the procedure used to protect the registry each time you add a new node to the network. Procedures 4–3 and 4–4 provide instructions for adding and deleting registry sites. Procedures 4–5 and 4–6 instruct you how to copy and move registries.

To add and delete registry sites, use the **crrgy** command with the **–a** or **–d** option. **Crrgy** adds or deletes site names to the master registry file, depending upon which option you use. **Crrgy** also has the capability of adding sites to the *rgy_master* file without accessing any of the sites (**–exist** option). This option can be useful when you are preparing to create a site on a node that is not yet available. The **–d** site option never accesses the deleted sites (in case they are unavailable when the command is issued). Instead, **crrgy –d site** deletes the site name from the *rgy_master* file. Refer to the *DOMAIN System Command Reference* for the **crrgy** options.

If you execute the **invol** utility on a site node, or change the node's name, or replace the node's disk, follow the procedures in Chapter 2 for maintaining root directories. Use the procedures described for **ctnode** or **ns_helper**, depending on how you have chosen to maintain root directories at your site. A registry is an upper–level directory and, as such, is subject to the rules of the network naming structure. Maintain site node entry directories as you would any other entry directory.

PROCEDURE 4–3: Adding a Registry Site

1. Add a registry site by executing the following command at the master site; type

   $ crrgy –a //new_node/registry/rgy_site[n]<RETURN>

   This command creates a new registry site at **new_node** and updates the file */registry/rgy_master*. It does not update */registry/registry* anywhere in the network.

2. After you have added the new registry site, update all node */registry/registry* copies of the master file *rgy_master*, by typing

   $ cpf /registry/rgy_master //alpha/registry/registry<RETURN>
   $ cpf /registry/rgy_master //beta/registry/registry<RETURN>

END OF PROCEDURE 4–3

PROCEDURE 4-4: Deleting a Registry Site

1. Delete a registry site by executing the following command at the master site; type

   $ crrgy -d //node/registry/rgy_site[n]<RETURN>

   This command deletes only the pathname *//node/registry/rgy_site[n]* from the file */registry/rgy_master*. It does not delete the directory *//node/registry/rgy_site[n]*.

2. After you have deleted a registry pathname from *rgy_master*, delete the directory *rgy_site* by typing

   $ dlt //node/registry/rgy_site[n]<RETURN>

3. Update all node */registry/registry* copies of the master file *rgy_master* by entering

   $ cpf /registry/rgy_master //alpha/registry/registry<RETURN>
   $ cpf /registry/rgy_master //beta/registry/registry<RETURN>

END OF PROCEDURE 4-4

PROCEDURE 4-5: Copying Replacement Master Registry

This procedure does not create any new sites. This example assumes you are using the same sites and moving only the file */registry/rgy_master*.

1. Create a replacement master registry file on the node you're working at. Type the following on a single shell command line:

   $ crrgy -ex -s //node/registry/rgy_site1 //node/registry/rgy_site2<RETURN>

2. Update all node */registry/registry* copies of the master file *rgy_master* by entering

   $ cpf /registry/rgy_master //alpha/registry/registry<RETURN>
   $ cpf /registry/rgy_master //beta/registry/registry<RETURN>

END OF PROCEDURE 4-5

PROCEDURE 4-6: Moving Master Registry

The following commands issued from *//newmstr* will move a registry from *//oldmstr* to *//newmstr* and add two additional sites.

1. Create a new *rgy_master* file without creating data files by entering the following command on a single shell command line:

   $ crrgy -r //nmstr exists -s //newmstr/registry/newsite1 //node2/registry/newsite2 //node3/registry/newsite3<RETURN>

2. Move existing data files to the first new site by typing the following command on a single shell command line:

   $ cpf //oldmstr/registry/oldsite1 //newmstr/registry/newsite1<RETURN>

3. From the *//newmstr* node, replicate *rgy_master* on *//newmstr* by typing

   $ cpf /registry/rgy_master /registry/registry<RETURN>

4. From the *//newmstr* node, move the data files to other new sites by typing

   $ salrgy<RETURN>

5. Distribute the new copy of *rgy_master* to every node in the ring by typing

   $ cpf //newmstr/registry/rgy_master //?*/registry/registry<RETURN>

   NOTES:

   The registry files at the old sites remain untouched.

   There may be difficulties doing this procedure if nodes are unavailable. Nodes will need to be updated when they are back on the ring.

   Additional difficulties may be encountered due to ACLs either on *//newmstr* or on other nodes. The registry system cannot violate ACL protections and it is the system administrator's responsibility to ensure that ACLs are correct for correct operation of the system.

END OF PROCEDURE 4-6

Maintaining Registry Database Consistency

To monitor and maintain registry sites, use **lrgy** (LIST REGISTRY) and **salrgy** (SALVAGE REGISTRY). The system automatically verifies database consistency whenever you or other users write to the registry. (You write to the registry when you change it by using **edacct** or **edppo**. Users write to it during login if they change their passwords or home directories.) The system performs its own version of **salrgy** if it finds database discrepancies. Execute **salrgy**

- Whenever you are unsure of registry database consistency
- Whenever a registry site that was not on the network during a network registry update comes back to the network.

Maintaining the Database

Network configuration changes may require that you move the registry database locations. Staff and security changes will require that you add, delete, and change user accounts. Use registry administration commands to update and maintain the registry database. Table 4–4 summarizes these commands. See the *DOMAIN System Command Reference* or online help files for detailed descriptions of each command.

To change the registry database, use **edppo** and **edacct**. Add and delete names in the ppo files, and change users' accounts, home directories, or passwords as necessary. Note that the master site and all other registry sites must be available when you use **edppo** or **edacct**. If the sites are unavailable, the system gives an error message stating that the registry update is only partially complete. If you edit ppo **or** account **files** after receiving this message, execute **sarlgy** when the sites are back in the network. Also, back up the registry files promptly whenever you change the user account database.

You can change any password with **edacct** even if you do not know the account's current password. Users may change their passwords and home directories at login. These changes are recorded automatically in the account file. If some registry sites are unavailable, users also receive the "partial update" error message described above. See the *DOMAIN System User's Guide* for information about changing passwords or home directories at login. The system maintains passwords in one-way encryption after they are entered in the account file. If a user forgets a password, a system administrator can assign a new one with **edacct**.

The shell command **chpass** (CHANGE PASSWORD) changes passwords from the command line. **Chpass, edacct, edppo,** and several other shell commands belong to a structure called the "login" protected subsystem. Chapter 5 discusses protected subsystems in detail.

Adding New Nodes to the Network

Whenever you add a new node to the network, first catalog the node as described in Chapter 2. Then use Procedure 4–2 earlier in this chapter to copy the registry files and directories to the node.

Setting ACLs on netsvc

The default ACL for **netsvc** is shown in Figure 4-11. It gives execute rights (x) to any user.

```
$  acl /com/netsvc -all
Acl for /com/netsvc:
  %.%.%.%                              pgndwrx
```

Figure 4-11. ACLs on netsvc

You can change the ACL on **netsvc** on nodes containing registries so that only a *sys_admin* account can have execute rights. The *DOMAIN System User's Guide* describes how to set ACL entries. This will protect the registry from careless or inexperienced users working directly at nodes that contain registries.

Merging Registries When Joining Networks

When you join two previously separate networks, you must merge the registries of the two networks into one registry for the entire internet. Before you join the networks, compare node names. If there are nodes with the same names in each network, change one name in each pair of duplicates before you join the networks. Procedure 4-7 shows how to compare and change node names.

PROCEDURE 4-7. Comparing and Changing Duplicate Node Names

1. In each network that you intend to join, execute **/com/lcnode** to generate a list of node names. For example in one network, type

   $ lcnode

   ```
   The node ID of this node is baab.
   3 other nodes responded.

     Node ID  Boot time          Current time      Entry Directory
     46FB  1987/06/10 10:26:56   1987/06/18 18:16:18  //bluegrass
     2009  1987/06/18 16:37:01   1987/06/18 18:09:59  //clover
     E18   1987/06/13 13:11:43   1987/06/18 18:15:57  //rye
   ```

   and in another network, type

   $ lcnode

   ```
   The node ID of this node is aabb.
   4 other nodes responded.

     Node ID  Boot time          Current time      Entry Directory
     2F17  1987/06/18 16:48:49   1987/06/18 18:11:46  //wheat
     C7C   1987/06/18  8:00:05   1987/06/18 16:21:59  //barley
     EDA   1987/06/10 10:21:50   1987/06/18 17:53:23  //heather
     2E22  1987/06/18 16:48:49   1987/06/18 18:11:46  //rye
   ```

2. Compare the lists for duplicate node names. In the example shown in Step 1, two nodes have the same name.

   ```
   E18   1987/06/13 13:11:43   1987/06/18 18:15:57  //rye
   2E22  1987/06/18 16:48:49   1987/06/18 18:11:46  //rye
   ```

3. Change the name of at least one of the nodes. For example, to rename one of the nodes currently named `rye` to `millet`, perform the following sequence of commands:

```
$ uctnode rye
$ ctnode millet 2e22 -r
$ ctnode millet 2e22 -r -on //?*
```

4. Install and activate whatever accessories (e.g., the DOMAIN Fiber Optic Link, DFL-100) that you are using to connect the two networks physically.

5. In the new network, update each node's root directory. At each node, type

```
$ ctnode -update
```

END OF PROCEDURE 4-7.

Comparing Registries

Each secure network that you joined has a separate registry structure. Once you've made a single network, you must identify and change any duplicate entries in the registry database: the ppo and account files.

Compare registries whenever you join secure networks into a single network. To compare registries, use the **cmppo** and **cmacct** commands according to the following format:

```
$ cmppo rgy1_path rgy2_path

$ cmacct rgy1_path rgy2_path
```

These commands allow you to specify any of the following registry pathnames:

- *//node/registry/rgy_master*. This file contains the name of the master registry file and the names of all site directories in the network. The file should list all current registry sites.
- *//node/registry/registry*. If you specify any node's copy of the master registry, **cmppo** and **cmacct** use the master registry listed in the node's */registry/registry* file. Use **lrgy** to see the contents of a node's copy of the master registry, since the local file can be out of date.
- *//node_name*. If you specify any node entry directory, **cmppo** and **cmacct** use the master registry listed in the node's */registry/registry* file. Use **lrgy** to see the contents of a node's copy of the master registry, since the local file can be out of date.

Table 4-5 lists the commands for comparing registries.

Table 4-5. Commands for Comparing Registries

Command	Purpose
cmacct	Compares the contents of account files and reports accounts that are associated with different home directories and passwords. Use **edacct** to resolve collisions.
cmppo	Compares the contents of ppo files and reports names that appear in both sets of ppo files and are associated with different UIDs. Use the **edppo** change command to change one of the names.
- pers	Compares person files only
- proj	Compares project files only
- org	Compares organization files only
- all	Compares all files

When you join two or more networks, each network had its own registry. You must identify and change any duplicate entries in these registries.

If the same name appears in two or more sets of ppo files, you must change the name in one of the registries. For example, if the name "Smith" appears in both registries' person files, use the **edppo** change command to change the name in one of the registries. Do *not* delete the name in either registry. Every name in the ppo files has a unique ID. The ID is included in the ACLs for objects that a user creates. If you delete rather than change names, users may be unable to access objects they've created.

If the name "Smith" appears in both person files, you must replace one instance of "Smith" with another name; for example, "JSmith." When a person name belongs to a single user who decides to keep one account and delete the other account, the user must change the ACLs on objects created by the account that will be deleted. The new ACLs must allow access from the account that will be kept.

NOTE: If **cmppo** finds duplicate names, make all the changes to one registry. Whenever you change any ppo files, you may outdate information in local registries for nodes that used those ppo files. Thus, if you change ppo files at one registry only, you minimize the number of nodes whose local registries are affected.

PROCEDURE 4-8. Comparing ppo Files

Perform the following steps to compare two registries.

1. Use the **cmppo** command with the **-pers** option to compare the person files for each registry.

```
$ cmppo //tulip/registry/rgy_master //bear/registry/rgy_master -pers

Names with different ids found in both registries PERSON file:
martin              smith

          2 names
```

2. If **cmppo -pers** finds duplicate person names, use **edppo -r** to make changes to *one* registry's person file.

```
$ edppo -r //tulip/registry/rgy_master -pers
=> l
martin    root      smith     sys_person     user
=> c martin bmartin
=> c smith msmith
=> l
bmartin   msmith    root      sys_person     user
=> wr
```

3. Repeat Step 1 to verify the change you just made to the person file.

```
$ cmppo //tulip/registry/rgy_master //bear/registry/rgy_master -pers

No names with different ids found in both registries PERSON file
```

4. Use the **cmppo** command with the **-proj** option to compare the project files. As with the person file, do *not* delete duplicate names. Change them with **edppo** as shown in Step 2.

```
$ cmppo //tulip/registry/rgy_master //bear/registry/rgy_master -proj
```

5. Use the **cmppo** command with the **-org** option to compare the organization files. As with the person and project files, do *not* delete duplicate names. Change them with **edppo** as shown in Step 2.

```
$ cmppo //tulip/registry/rgy_master //bear/registry/rgy_master -org
```

6. Repeat Steps 1 to 5 as necessary; this depends on how many networks' or internets' registries you join to form a single registry.

END OF PROCEDURE 4-8.

Perform Procedure 4-9 if you made changes to ppo files. When you make changes to ppo files, local registries on some nodes may contain outdated information that will not be refreshed automatically. The outdated information will not affect most users' ability to log in. However, any user whose ppo information changed can lose the ability to log in if either of the following conditions is true:

- The node cannot access any updated site directories.
- The node contains outdated information in its local ppo files.

To prevent problems, delete and recreate local registries that may contain outdated ppo files. If you know which nodes contain outdated ppo files, fix the local registries at those nodes. For example, if you changed a person name from "martin" to "bmartin," you can delete and recreate the local registries at the node(s) that "bmartin" uses. However, if you cannot identify the nodes that may be affected by the changes, (if you change project or organization names this is very likely), delete and recreate the local registries at *each* node that uses the registry site that contains new ppo files.

PROCEDURE 4-9. Fixing Local Registries

Use the following commands to delete and recreate a local registry:

```
$ dlf //node/registry/local_registry
$ dlt //node/registry/local_site
$ crrgy -loc -r //node
```

END OF PROCEDURE 4-9.

Compare the registry account files. If there is a duplicate entry, use **edacct** to delete one of the accounts. It is very likely that accounts like user.none.none will be duplicates. Procedure 4-10 gives you the steps for comparing these files.

PROCEDURE 4-10. Comparing Account Files

1. Use **cmacct** as shown. Type:

```
$ cmacct //tulip/registry/rgy_master //bear/registry/rgy_master
Account user.none.none has different home directories:  /   //guest/use
        1 account collision
```

2. If **cmacct** finds duplicate accounts, use **edacct -r** to make changes to *one* registry's account file.

```
$ edacct -r //tulip/registry/rgy_master
=> d user none none
user      none      none      /
OK to delete this account? (Y/N/Quit):  y
Current entry is:   tsmith     pluto     r_d    //daisy/smith
=> wr
```

3. Repeat Step 1 to verify the change you just made to the account file.

```
$ cmacct //tulip/registry/rgy_master //bear/registry/rgy_master

          No account collisions
```

4. If you changed any node names when you joined networks (Procedure 4-7), change the pathnames of home directories on any nodes that received new names. In the example we used, the node 1231.D89, named *//lily* became *//tiger*. The account file that contains this information is on *//bear*.

```
$ edacct -r //bear/registry/rgy_master
=> la
martin    pluto      r_d     //lily/martin
davis     plato      r_d     //dog/davis
fred      plotter    r_d     //lion/fred
jim       sys_admin  none    //bear/jim
smith     sys_admin  none    //bear smith
=> d martin
=> a martin pluto r_d //tiger/martin
=> wr
```

5. Repeat Steps 1 to 4 as necessary; this depends on how many networks' or internets' registries you join to form a single registry.

END OF PROCEDURE 4-10.

Merging Registries

After you have changed duplicate entries, use the **mrgrgy** command to merge the registries from each of the networks. The **mrgrgy** command

- Selects one file to be */registry/rgy_master* for the entire network and merges the contents of all master registry files from each formerly separate network into the network master. Only one master registry is retained, and it contains all the information that was in each network master registry.

- Merges the contents of all site directories' ppo and account files, so each site directory contains information on all users in the network.

The **mrgrgy** command format

mrgrgy *rgy1_path rgy2_path*

allows you to specify rgy_path using any of the following:

- *//node/registry/rgy_master*. This file contains the name of the master registry file and the names of all site directories in the network. The file should list all current registry sites.

- *//node/registry/registry*. If you specify any node's copy of the master registry, **cmppo** and **cmacct** use the master registry listed in the node's */registry/registry* file. Use **lrgy** to see the contents of a node's copy of the master registry since the local file may be out of date.

- *//node_name*. If you specify any node entry directory, **cmppo** and **cmacct** use the master registry listed in the node's */registry/registry* file. Use **lrgy** to see the contents of a node's copy of the master registry since the local file may be out of date.

The file you specify in *rgy1_path* becomes the new master registry file. The **mrgrgy** command merges the contents of *rgy2_path* into *rgy1_path* so that *rgy1_path* becomes the merged master. *Rgy2_path* is deleted.

Although Figure 4–12 represents the action of **mrgrgy** in a general way, the command actually completes in four phases. The output from the command indicates the phase that is currently executing. Some events; for example, network failures, may cause the **mrgrgy** command to abort before it completes the merge operation. There is a recovery action for phases that abort. Table 4–6 describes each phase and the appropriate recovery action.

Plan to merge registries at a time when network traffic is light, and when all the site directories on all networks are available. The **mrgrgy** command copies data within each network. In a large network, it may take a while to merge registries.

Note that, after you merge registries, you must also copy the merged master to each node's */registry/registry*. Then, each node will know the names of all site directories in the network.

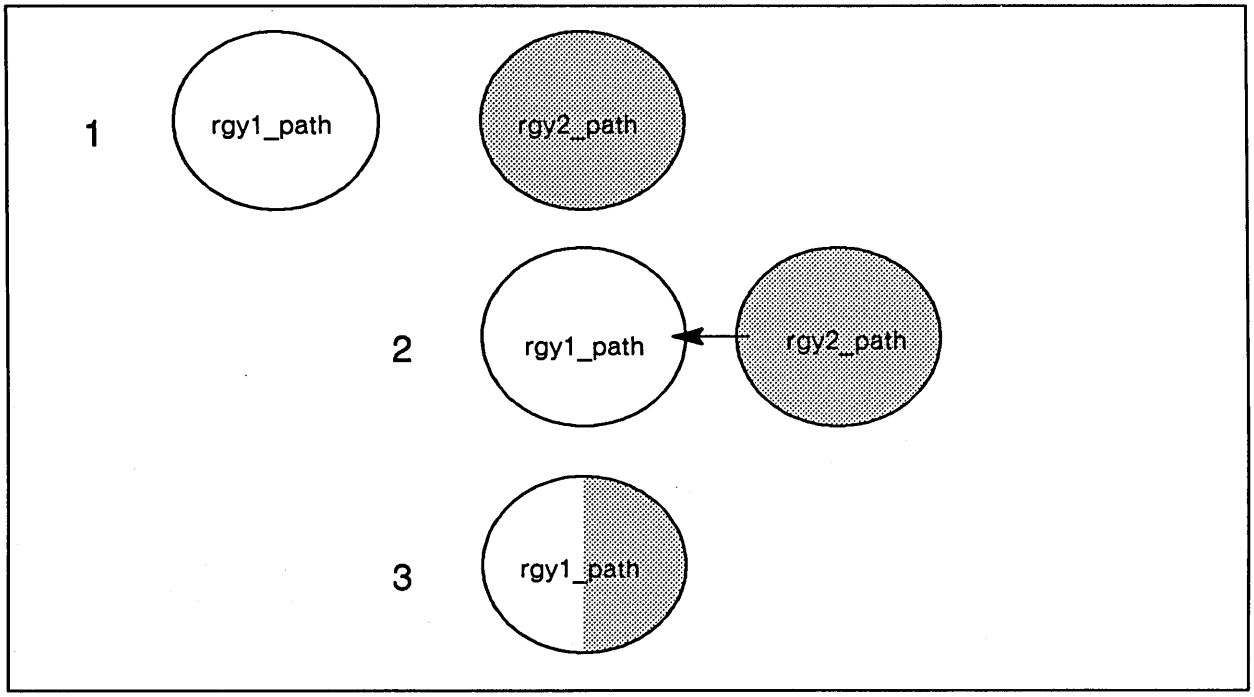


Figure 4–12. Action of **mrgrgy** *on /registry/rgy_master*

Perform the merge by selecting one node to become the master registry for the entire network. If you are joining separate networks, it does not matter which master you select. If you add networks to an existing network, select the existing master registry.

Use Procedure 4–11 to merge each of the registries you compared, *one by one* into the master registry. In the procedure, *//tulip/registry/rgy_master* becomes the master registry and *//bear/registry/rgy_master* is deleted. During the merge, the contents of all site directories that exist on the networks are merged into *//tulip/registry/rgy_site*.

Table 4-6. Phases and Recovery Actions for mrgrgy

Phase	Description
1	**mrgrgy** creates new, merged ppo and account files and saves them in temporary files. Some .bak files are changed but no changes are made to the original ppo and account files. Failure modes: **mrgrgy** detects duplicate entries in ppo or account files. Repeat all procedures. Rerun. **mrgrgy** fails due to network problems. Rerun.
2	**mrgrgy** saves the ppo and account files it created in Phase 1 in the site directories listed in rgy1_path. Failure modes: **mrgrgy** fails due to network problems. Rerun.
3	**mrgrgy** creates */registry/rgy_master* for the internet and saves it in rgy_1 path. **mrgrgy** *deletes rgy2_path.* Failure modes: **mrgrgy** fails due to network problems. Recreate original rgy1_path and rgy2_path from their .bak files. Rerun **mrgrgy.**
4	**mrgrgy** executes **salrgy** to copy the ppo and account files created in Phase 1 to all site directories listed in the merged master created in Phase 3. Failure modes: **mrgrgy** fails due to network problems. Execute **salrgy.**

PROCEDURE 4-11. Merging Registries

1. On each network make a copy of the master registry file and at least one site directory. If the node with the master registry file also has a site directory, you can use the command, **cpt** (COPY_TREE) to make backup copies.

```
$ cpt //tulip/registry  //tulip/save_registry
$ cpt //bear/registry  //bear/save_registry
```

2. Execute **mrgrgy** by specifying the registry that will become the (rgy1_path) first; then specify the other master (rgy2_path).

```
$ mrgrgy //tulip/registry/rgy_master //bear/registry/rgy_master

Phase 1:

Merging the PERSON files
Merge completed

Merging the project files
Merge completed

Merging the org files
Merge completed

Merging the full names files
Merge completed


 Merging the account files
 Merge completed


 Phase 2:

 Merged PERSON file saved in registry //tulip/registry/registry's sites

 Merged project file saved in registry //tulip/registry/registry's sites

 Merged org file saved in registry //tulip/registry/registry's sites

 Merged full_names file saved in registry
  //tulip/registry/registry's sites

 Merged account file saved in registry //tulip/registry/registry's sites
 Phase 3:

 Creating the new registry's master file in "//tulip/registry/rgy_master"
 New master file completed


 Phase 4:

 Merged PERSON file saved in all new registry's sites

 Merged project file saved in all new registry's sites

 Merged org file saved in all new registry's sites

 Merged full_names file saved in all new registry's sites

 Merged account file saved in all new registry's sites
```

3. Repeat Step 2 until all registries have been merged into the internet master.

END OF PROCEDURE 4-11.

You must update each node's copy of */registry/registry* after you merge the old registries into a new master registry. Perform Procedure 4–12 to update */registry/registry* on each node in the network. Copy the master registry file to each node's copy of */registry/registry*.

PROCEDURE 4–12. Update Node Copies of Registry

Update each node on a network:

```
$ cpf //tulip/registry/rgy_master //alpha/registry/rgy_master -r
```

END OF PROCEDURE 4–12.

Chapter 5

Protecting System Registries and Software

The network registry allows the operating system to control user access to the network. You will use the methods described in this chapter to protect system and user software on each node. You will also perform regular backups of system and user directories to protect work in progress. Use subsystems, Access Control Lists (ACLs), and Subject Identifiers (SIDs) to provide selective levels of security for parts of the system or directories. This chapter describes

- Protecting registries and system software on nodes
- Installing new DOMAIN software
- Creating system backups to protect user files
- Using subsystems, SIDs, and ACLs to selectively grant access to the node or network

We've included some information about ACLs which pertains to this discussion. For complete information about manipulating ACLs, as well as about protected subsystems and SIDs, see the *DOMAIN System User's Guide*.

ACLs

The ACL is the DOMAIN system's method of supplying file and directory protection. ACLs are actual objects that define who can perform operations on the file or directory and what operations particular groups of users can perform. ACLs take up space on your disk (at least one block each), so there is incentive to keep the number of ACLs on your system to a minimum.

Normally, the DOMAIN system shares ACLs by means of the **initial file ACL** and the **directory ACLs**. A newly created file's ACL is set according to the initial file ACL of the directory in which it is created, and certain forms of access to the new file are controlled by the directory's other ACLs.

Briefly, an ACL entry consists of two parts: the (SID) and the Access Rights. The SID contains four fields, each of which contains a 64-bit identifier for person, project, organization, and node,

abbreviated **ppon**. The format of the Access Rights depends on whether the object being protected is a file or directory; these formats are discussed individually below. ACL entries are sorted from most specific to least specific. (If two ACL entries have only one field named and the other fields specified with percent characters (%), the leftmost field is sorted first.) For example,

```
user.r_d.%.%
```

would be interpreted before

```
user.%.%.%
```

which, in turn, would be interpreted before

```
%.r_d.%.%
```

It is the combination of all ACL entries that makes up the ACL for an object.

The ACL for a File

A DOMAIN file has one ACL that consists of a number of individual ACL entries. The Access Rights scheme for each ACL entry allows or denies the following permissions:

- **p** Protect: change the file's ACLs
- **g** Grant permission to others
- **n** Change nodes from which users can access files
- **d** Delete permission
- **r** Read permission
- **w** Write permission
- **x** Execute object files

The ACLs for a Directory

Each directory on the system has ACL entries, as well. However, a directory is an object that may contain other objects. It may have a different ACL for the directory itself, for files created in the directory, and for subdirectories created under the directory. (Each subdirectory then has three ACLs associated with it, and so on.) The two types of ACL entries that pertain to directories take the same form, which is specified in the subsection "The Directory ACL." The third type, which affects files created under the directory, takes the same form as the ACL for a file.

The Directory ACL

The directory ACL controls the access to the directory itself. Each entry in the ACL consists of the following Access Rights:

- **p** Change the directory's ACLs
- **g** Grant permission to others
- **n** Change nodes from which users can access the directory
- **d** Delete the directory
- **c** Change names and delete links

a Add files and subdirectories

l Add links

r List entries

s Access subdirectories and subdirectory objects

e Delete subdirectories and subdirectory objects

Initial Default File ACL

The initial default file ACL is the ACL that is assigned to files created in the directory. Entries in this ACL have a format identical to those in the file ACL. Normally, an AEGIS program that creates a file uses the initial default file ACL.

Initial Default Directory ACL

The initial default directory ACL is the initial ACL assigned to subdirectories created in the directory. The format of its entries is identical to the format of entries for the directory ACL.

Protecting Registries and System Software

Every object in the system has an ACL that specifies the rights (read, write, delete, etc.) a user has to that object. Secure networks contain registries that control users' access to network resources. In an open network, one without registries, every user has complete rights to every object. When you receive new nodes with disks, the ACLs for objects on the disk make it an open network node.

After you create a registry, the network will not be secured entirely until you execute a **protection program** supplied with the DOMAIN system software. The program protects system software and all parts of the registry residing on each node. The program simplifies the task of assigning a complex set of ACLs at each node. This section describes the levels of protection available for the nodes in the network. Procedure 5-1 describes how to execute the protection program.

Execute the protection program at each node when you first set up the network. The program changes the default ACLs on a node to be consistent with the level of protection you select. Whenever you add another disked node to the network, decide on the level of protection the node should have and execute the program. When you receive a new software release, the protection program is included automatically by the software installation procedures.

Selecting the Level of Protection

The protection program uses two files to protect system software: *sys_acls* and *sys_idacls*. The files are created in the utility directory, */install/acl_dir*. The contents of these files come from a set of **ACL templates** also supplied with your system. ACL templates are lists of ACL entries and initial default ACL entries for system files and directories, including registry files and directories. The protection program reads *sys_acls* and *sys_idacls* and assigns the ACLs listed in these files.

When you execute the protection programs on nodes with newly created registries, and on newly installed nodes, you select ACL templates. When you install a new DOMAIN software release, the software installation procedures automatically copy the ACL templates you select into the files.

ACL templates are based on the following assumptions about network users:

- General users simply use a node and have no administrative privileges or responsibilities. ACLs can prevent these users from changing or deleting any system software.
- Node administrators are responsible for the proper operation of a particular node(s). ACLs can grant them access to libraries, the operating system, and local commands.

- System administrators are responsible for the overall security of a network. ACLs can grant them full access to all parts of the system, including registries, protected subsystems, and system commands, that are not available to the node administrator.

ACL templates let you assign different sets of ACL entries to different nodes in the network. For example, you can arrange that on some nodes any user has access to system software, but on other nodes only system administrators have this access. For this reason, three categories of ACL templates are available. There are two templates in each category. One template assigns ACLs, and the other assigns initial default ACLs. If the ACLs assigned by the templates do not match your needs, edit either *sys_acls* or *sys_idacls* to modify the ACLs to fit your requirements.

The ACL template categories create the following types of nodes. Table 5–1 shows the pathnames of ACL templates in each of the three categories.

- *Open_node* has the minimum protection for a secure network. On nodes protected with this option, general users (including "user.none.none") have the same rights to much of the system software as system administrators. These templates withhold only rights that might affect the system's ability to operate correctly. For example, users can't delete system software or registries.

- *Personal_node* has moderate protection. Node administrator privileges are limited to a single user or group of users. A system administrator is required to perform some functions.

  These templates offer moderate protection for a secure network. In them, you specify a node administrator who has more rights than general users, but fewer rights than system administrators. As with open node templates, the personal node templates prevent anyone from deleting pieces of system software, as protection for the network. Also, they allow only system administrators access to security–related files and directories.

- *System_node* has the highest protection. Only system administrators can perform any administrative functions on this node. Of the three categories, these templates provide the greatest protection.

Table 5–1. ACL Template Filenames

Node Type	Template Filenames (all in */install/acl_dir*)
open_node	secure_net_open_node_acls secure_net_open_node_idacls
personal_node	secure_net_personal_node_acls secure_net_personal_node_idacls
system node	secure_net_system_node_acls secure_net_system_node_idacls

No matter what protection category you choose, you can make some kinds of safe modifications to your registry, yet maintain overall registry security. These modifications are described in the section entitled "Protected Subsystem Status."

Reading ACL Templates

To see the entire set of ACLs the protection program assigns to nodes, display any of the ACL template files listed in Table 5–1 on a monitor screen. The template will serve as a reference when you read Table 5–2.

The top of each ACL template file contains some comment lines. Scroll through the comment lines to a table consisting of the pathnames of system files, directories, or wildcard specifications. The ACL entries assigned to the listed objects appear at the right. The pathname and ACL entry has the format shown in Table 5-2.

Table 5-2. Fields in ACL Templates

Column Number	Contents
1-28	Pathname of file or directory; must start with '/'; may be wildcard
29	'f' or 'd' for File or Directory
33-40	Rights assigned to %.%.%.% (anyone) All rights in the table are assigned in the form used by the edacl command (see the DOMAIN System Command Reference).
45-52	Rights assigned to %.sys_admin.%.% (any system administrator).
57-64	Rights assigned to additional SID class (#1) (may be left blank). When you use the personal_node templates, these rights are for node owner, the person responsible for the node, and are already assigned.
69-76	Rights assigned to additional SID class (#2) (may be left blank or filled in). The header for this column is "PPO2."

The following line from a sys_acl template in the secure_net_personal_node category shows the ACL assignments for the */sys/help* directory.

```
                    % rts           %.sys_admin    PPO1

/sys/help           d   r           pgndcalr       pgndcalr

                    READ and
                    DELETE
                    rights for      All rights for %.sys_admin.%.%
                    %.%.%.%         and for the person responsible
                                    for this node.
```

If you execute the protection program with the *secure_net_personal_node* category, give the name of the node "owner" or administrator, for example, Fred. The ACL for the */sys/help* directory shows the result of this *sys_acl* specification. Fred.%.lab.% gives full rights to the node administrator and a system administrator, but not to any user.

```
$ acl /sys/help
Acl for /sys/help:
   fred.%.lab.%                 pgndcalrse
   %.sys_admin.%.%              pgndcalrse
   %.%.%.%                      ---d---r
```

Editing ACL Templates

To edit a copy of ACL templates, the protection procedure directs you to copy the templates you choose into *sys_acls* and *sys_idacls*. Edit these files if you want to

- Change the rights listed in the ACL templates
- Specify additional rights for SID classes 1 and 2 (if you chose *open_node* or *system_node* templates)
- Specify additional rights for SID class 2 (if you chose *personal_node* templates)

The following paragraphs provide some general guidelines for editing ACL templates.

When you edit ACL template copies, do not eliminate any of the rights assigned to *%.%.%.%* and *%.sys_admin.%.%* in the template. Eliminating any of these rights could affect the operation of your system software.

The bottom portion of an *open_node* template contains files and directories that affect network security. The rights specified in the template for these objects are the maximum allowed for secure system operation. Granting more rights than those specified can make your system insecure.

If you specify rights for each additional SID class listed in Table 5–2, be sure to use the correct columns. The template lists the column numbers.

Specify the same rights that you specify with the **edacl** command. For files, specify **pgndrwx** or a subset thereof (use a hyphen (–) in place of any right you wish to withhold). For directories, specify **pgndcalrse** or a subset thereof (use a hyphen in place of any right you wish to withhold).

Whenever you specify your own name (as it appears in *rgy_site[n]/person*, in one of the additional SID classes, give yourself p–rights. The ACL creation will fail if you don't authorize yourself to change ACL entries.

ACL templates do not set ACL entries for the node's entry directory (*//*). Assign ACLs to node entry directories manually. Preface comments in the template file with the comment character (#). The protection program ignores the comment character and everything that line follows it on the line.

Running the Protection Program

This section provides a procedure for running a program that protects your registry database and system software in a secure network. Use Procedure 5–1 when you

- Create a registry for the network
- Create a registry for a new node in an existing network
- Change the location of *rgy_master* or *rgy_site[n]* and the new location does not have adequate security.

When you install new software in a secure network, the software installation procedure automatically executes the protection program.

Run the protection procedure on any disked node in the network. Before executing the procedure, use the **lvolfs** command to determine the amount of disk space available on the node. The procedure needs 3000 blocks of disk space. If the disk space is inadequate, remove user files, not DOMAIN system files, to perform the procedure. When the procedure is finished, return the files to the disk.

Run the procedure first on the master site node. Then, repeat it on each of the other site nodes and, finally, on each nonsite node. The master site node must be available when you run the procedure on other nodes.

If a node's system software contains links to other system objects, the protection procedure sets the ACLs for the object to which the link points. For example, if */sys/help* on one node is a link to the

/sys/help tree on another node, the protection procedures will set ACLs for the second node's */sys/help* tree.

In a new network, execute the protection procedure immediately after creating the registry and before creating any links in place of standard software. Then, you need not be concerned with links when you run the protection procedure.

If there are links on nodes, of the type described above, delete the links before executing the procedure, then re-create them when the procedure is finished.

NOTE: Before executing this procedure, the node must be cataloged and must have its registry files and directories.

Procedure 5-1 illustrates responses to the protection program prompts. Give responses appropriate to your site when you execute this program. The output shown illustrates the "most complicated case" (i.e., that there are customized ACLs for the node). If you select an ACL template from the ones supplied, there are fewer prompts than are illustrated here.

PROCEDURE 5-1: Protecting the Registry Database

1. Log in with a *sys_admin* account. Set the working directory to */install* and invoke the install procedure:

```
$ wd install
$ install<RETURN>

Software Installation Types are:

STD        -- Install SR9 standard software
RESTART    -- Restart the software installation
OPT        -- Install optional software
ACL        -- Set acls for existing software
CLEANUP    -- Run the cleanup procedure for ADD MODE installations
DOMAIN/IX -- Install the DOMAIN/IX software

Please enter Installation Type: acl<RETURN>
You are logged in as:
    person.sys_admin.organization.node  //node_name
Do you have adequate rights?
Please enter response. (yes or no) yes<RETURN>
Please enter //target_node name: //george<RETURN>

ACL Template Types are:

OPEN      --    This type of node has the lowest protection. Any user
                can perform node administrator functions. A system
                administrator is required for some functions.

PERSONAL  --    This type of node has moderate protection. Node admin-
                istrator privileges are limited to a single user or
                group of users. A system administrator is required to
                perform some functions.

SYSTEM    --    This node has the highest protection. Only system admin-
                istrators can perform any administrative functions on
                this node.

USER      --    Choose this option if you have saved your ACL templates
                from a previous installation or update and wish to rein-
                stall those ACLs. You must copy your own customized
                version of the ACL templates to the files //node_name/
                install/acl_dir/sys_acls and //node_name/install/
                acl_dir/sys_idacls before you can continue.

Please enter the type of ACL template you would like: user<RETURN>

Have you already copied your customized version of the ACL and IDACL
templates to the files:
```

```
    //george/install/acl_dir/sys_acls
    //george/install/acl_dir/sys_idacls
 Please enter 'YES' or 'NO' : yes<RETURN>
```

If you enter "NO," the script prompts as follows:

```
Please copy your own customized version of the ACL  template  to the files
sys_acls and  sys_idacls.   Do this by typing:

$ cpf customized_acls //george/install/acl/acl_dir/sys_acls -r -l
$ cpf customized_idacls //george/install/acl/acl_dir/sys_acls -r -l

Then you must RE-START this procedure.

If you have already copied your customize ACLs to the target node the
script will continue as follows:

Have you specified an additional PPO?
Please enter 'YES' or 'NO': yes<RETURN>

If you answer "YES" the script will prompt you for the PPOs as follows:

Please enter the first PPO: #enter additional PPOs here#

PREPARING ACLS
        .
        .
SETTING ACL'S
ACL'S CAN NOT BE SET FOR OBJECTS THAT ARE IN-USE.

NOTE: You may get the following error message:

?(acl) Acl not changed for   - object is in use (OS/file server)
This error is expected and will not affect the installation procedures.
          .
          .
          .
Finished Installing ACL's on //george
Please shutdown, reset and restart //george
```

2. Shutdown, reset, and restart the node.

END OF PROCEDURE 5-1.

Installing New DOMAIN Releases on Secured Networks

We only guarantee that consecutive software releases are compatible. Therefore, do not try to run a network with some nodes running SR7.0 or SR8.0 software and some running SR9.0 software. Software installation instructions are included in the release notes.

Only persons with a *sys_admin* account can initially install software. Refer to the *SR9.6 Release Notes* and *Installing DOMAIN Software* for the procedures to use to install new software on nodes. The software installation program uses the protection program (Procedure 5-1) to protect the new software and registry.

Backing Up Your System

To protect the information on nodes, copy files to some off-network storage media, such as magnetic tape, at regularly scheduled intervals. It is not necessary to copy every file, on every node, to tape. Save system software once. If you do not want to back up copies of system software at all, it is especially important to save the media shipped with the nodes.

Since many nodes in a network hold the same system software, be selective about the files that are backed up regularly. Each node contains, for example, the same */systest* directory. Save this directory from one node and restore it to other nodes as necessary.

One way to selectively save files is to create a back-up list on every node. Tell users to enter the names of upper-level directories that they want to save in the node's back-up list. Save system and server log files by entering their names in the back-up list. Then write a shell script that reads the back-up list and writes the directories in the list to magnetic tape. Use the **wbak** (WRITE BACKUP) command to write the files to magnetic tape.

Remember that the system cannot write a file to tape unless the ACL for the file allows Read access. If you write a shell script to create the system backups, be sure that your system's ACLs are such that the shell program can read all needed files.

The **wbak** command creates a file called *backup_history* in each directory that it backs up. The log-in account that executes **wbak**, directly or through a shell script, must have Add access to the directories being backed up and Write access to the *backup_history* file. See below for detailed information about the *backup* project account for people doing system backups.

You can restore the objects saved on tape to a destination directory with the **rbak** (READ BACKUP) command. By default, **rbak** assigns the destination directory's default ACL to the object being restored. So, by default, the restored file or directory has the same access control assignments as the directory to which it is restored. If you want the object to retain its original ACL, you must use the **-sacl** option with **rbak**.

Protected Subsystem Status

The operating system associates a Subject Identifier (SID) with each process the user creates from login to logout. For example, processes executed by the user's start-up file have SIDs. SID classes correspond to the registry **ppo** files. ACL rights are attached to SID classes.

The system gathers SID information during the log-in procedure. The first field in an SID contains a person name, the second field is the user's project name, the third field contains organization names. The information contained in these three fields is referred to as an SID class. The fourth field in an SID identifies the hexadecimal ID of the node where the user is working. The system assigns the same SID to every process a user creates during a log-in session.

An exception to the usual procedure described above occurs when the user executes the **login** command in the shell input window. In this case, the operating system starts a new log-in process and assigns a new SID to that process. The SID for this process also ends when the user logs off.

When a process requests access to a file, directory, program or another process, the system compares the process' SID with the ACL of the requested object. If the requested object's ACL includes an

SID specification that matches the process' SID, the system grants the process the access rights specified in the ACL.

Usually, a program can access a file if the process running the program has the proper SID. The ACL templates supplied with your DOMAIN system ensure that system server processes have SIDs and ACLs that will not hinder user access.

At times, you may want to grant a program access rights to certain data files, regardless of the program's process SID. The term "protected subsystem" refers to a set of data files and a program that has special access rights to the files. The *DOMAIN System User's Guide* describes how to create protected subsystems. In the following sections, we describe those aspects of protected subsystems of particular interest to system administrators.

The login Protected Subsystem

The protection program makes your registry a data object in the "login protected subsystem." The login protected subsystem allows users to change their own passwords or home directories while logged in, or to change SIDs while logged in. However, only system administrators can add or delete accounts, create new site nodes, or other operations that affect the security of the network at large.

The following list shows the shell commands that are part of the login subsystem.

- The **chpass** command allows users to change their password at the command shell level.
- The **chdir** command allows users to change their home directories at the command shell level.
- The **edacct** command allows system administrators to edit the account files.
- The **edppo** command allows system administrators to edit the person, project, and organization files.
- The **xsubs** command allows shell programs to run as subsystem managers.
- The **login** command allows users to change their SIDs. Using this command, users can log in to an existing process with a different SID. For instance, a user who originally logged in with the SID "nicholas.none.none.1cad" might need the more specific SID "nicholas.none.dickens.1cad" to have full rights to an object.
- The **siologin** command is part of the login subsystem that allows users to log in to a DOMAIN network via a modem or terminal attached to a serial I/O (SIO) line on a DOMAIN node. Chapter 6 describes **siologin**.

Refer to the *DOMAIN System Command Reference* for information about the shell commands listed above. The system runs several other login manager programs for use with DSPs.

These are the default ACLs on the login protected subsystem.

```
$ acl /sys/subsys/login
 Acl for sys/subsys/login:
 Subsystem login manager
   %.sys_admin.%.%                    p-n--rx
   %.%.%.%                            -----rx
```

Using Protected Subsystems to Grant Access Selectively

This section provides an example of how one system administrator used ACL entries and protected subsystems to protect system resources. The example shows how administration personnel at one site protected a set of node listing records.

The example is very simple, but there are more complex and powerful uses for protected subsystems. If you have sensitive programs that you want users to run only with certain options, you could write manager programs that run these programs and deny Execute rights to the sensitive programs themselves. You could do the same thing if you want the sensitive programs run only for certain purposes.

In our example, a system administrator's assistant, named Art, used the Netmain Interactive Tool to produce a weekly listing of all the nodes in the network. He named the file *node_list.date* (in yy.mm.dd format) and stored it in his *records* directory, a subdirectory of his home directory. Fran, the system administrator, wanted Art to sort some of the information and store the sorted files in her *confidential* directory. She did not want Art to have Read or Write rights to any of the files in the *confidential* directory. The following is a description of how she set up ACLs for these files and directories, including a transcript, shown in Figure 5-1.

Fran first created a protected subsystem called *node_list_protect*. She then created several links to her confidential directory and created a dummy file in this directory. She defined the dummy file ACL entries so that she was the only user with full rights, and then made it a *node_list_protect* data file. Then she wrote the file-sorting shell script and put a copy in Art's personal *com* directory.

The script starts with the **subs -up** command to increase privilege to allow it to act on subsystem objects. The script then creates several files. (The script also includes a command that sets ACLs to make these files into *node_list_protect* data files.) Finally, Fran made the script a *node_list_protect* subsystem manager. When Art wanted to use the sort program, he executed the following command:

$ xsubs node_list_protect *yy.mm.dd*<RETURN>

Figure 5-1 shows the transcript.

```
      #Create the subsystem.
 $ crsubs node_list_protect

      #Create home directory links to
      #//mc/fran/confidential.
      #Use links instead of the cumbersome pathname.

 $ crl -con //mc/fran/confidential
 $ cpl -con //mc/art

      #Create a dummy file in -con.
$ wd -con
$ crf node_list.dummy

      #Enter the node_list_protect subsystem.
$ ensubs node_list_protect

      # Give node_list.dummy data object status in node_list_protect.
$ subs -con/node_list.dummy node_list_protect -data

      # Leave the protected subsystem.
$ *** EOF ***
```

Figure 5-1. Sample Transcript

```
     # Make ACL entries for node_list.dummy.
 $ edacl ~con/node_list.dummy -cf fran.%.lab.% -owner
 $ edacl ~con/node_list.dummy -cf fran.sys_admin.lab.% -owner
 $ edacl ~con/node_list.dummy -af %.backup.%.% r
 $ edacl ~con/node_list.dummy -cf %.%.lab.% -none

     # Check the ACLs.
 $ acl node_list.dummy
Acl for node_list.dummy:
Subsystem node_list_protect object
  fran.%.lab.%                      pgndwrx
  fran.sys_admin.lab.%              pgndwrx
  %.backup.%.%                      -----r-
  %.%.%.%                           -------

     #Now write the shell script.
     #This is the text of the shell script that Fran wrote.
     #Her assistant uses it to sort node information.


     # Script:  Node_Sort_Program
     # Its pathname is //mc/art/com/node_sort_program.
     #
     # This script sorts nodes in the node list into
     # alphabetical order; then it sorts nodes by node ID.
     # This indicates relative age of the node.
     # Enter the node list name as nodes.^1, where ^1 is the
     # date in yy.mm.dd format.
     #
       subs -up
       srf -s 18 //mc/art/records/nodes.^1 >alphabetical_list.temp
       srf -s 10 //mc/art/records/nodes.^1 >id_list.temp
       catf alphabetical_list.temp id_list.temp >~con/list.^1
             # The next command copies the acl from the dummy program
             # to the sorted file created in this shell script.
       acl //mc/art/con/list.^1 //mc/art/con/node_list.dummy
       args "The file //mc/art/con/list.date contains 2 lists."
       args 'The first list shows the nodes in alphabetical order,'
       args 'The second list shows them by node ID (oldest to newest).'
       dlf alphabetical_list.temp id_list.temp
       subs -down
            # End of shell script


     #Re-enter the subsystem, give the shell script MGR status,
     #and check its status.
 $ ensubs node_list_protect
 $ subs //mc/art/com/node_sort_program node_list_protect -mgr
 $ subs //mc/art/com/node_sort_program
"...node_sort_program" is a NODE_LIST_PROTECT subsystem manager
"...node_sort_program" is a file subsystem data object


     #Leave the subsystem.
 $ *** EOF ***
```

Figure 5-1. Sample Transcript (Cont.)

Using SIDs to Grant Special Access

The registry site directory's project file contains the names *sys_admin* and *backup*. Use these names to grant access privileges that allow certain users to maintain the system.

To enable certain users to perform a system-wide backup, use the **edacct** shell command to create accounts with the project name "backup." Then, instruct users to define the following ACL entries for objects that they want backed up:

- Anyone with project name "backup" must have Read access to the object.
- Anyone with project name "backup" must have Write access to the *backup_history* file in the node's upper-level directory.
- Anyone with project name "backup" must have Add access to the directory one or more levels above the object. To enable a user to create or maintain the entire registry database, create accounts with the project name *sys_admin*. Then, grant users with this project name full access (except Delete rights) to the registry database.

The system assigns the default project name *server* to any processes created with the Display Manager command **cps** (CREATE PROCESS SERVER). This status is also assigned to DOMAIN system servers described in Chapter 6. Processes with the project name *server* are independent of log-in activity. Use the **cps** command to create processes that run regardless of log-in activity, for example, device driver routines. Do not assign the project name *server* to any accounts.

User.none.none Status

The registry site directory files contains an account, "user.none.none.," which enables anyone to use the network without special rights. You may use ACLs as described in the *DOMAIN System User's Guide* to limit the rights assigned to this SID.

Chapter 6

Network Servers

The first part of this chapter provides general information about servers. It includes discussions about the attributes of servers, as well as starting, stopping, and maintaining server processes.

The second part of this chapter provides reference material on each of the servers that arrive with DOMAIN system software. Server descriptions in this chapter are given in alphabetical order. The reference information contains

- A description of the server process
- Methods for starting, stopping, and reinitializing the server process
- Information on server configuration files
- Information on server options and arguments
- Examples of options in use
- Special considerations, if any, about the server processes
- References to other information that may be available on the server process

Two servers, **netmain_srvr** and **ns_helper**, have interactive tools. The **edns** utility, the system administrator's tool for use with **ns_helper**, is described in the *DOMAIN System Command Reference*. **Netmain**, the tool for use with **netmain_srvr**, is described in Chapter 10.

Note that this chapter differentiates between creating servers on user nodes (such as the DN4xx, DN6xx, or DN3xx) and DOMAIN Server Processors (such as the DSP80), because you use different methods to create servers on DSPs and on user nodes.

General Information on Servers

DOMAIN network server processes

- Manage user access to network resources, (e.g., printers)
- Manage requests for access to data and the transfer of data
- Gather statistics about the state of the network
- Manage communication pathways outside the network

When properly configured, server processes are transparent to network users, provide cost-effective use of expensive peripheral devices, and enhance the ability of users to share information resources.

Read both the general and particular information on server processes to decide how to realize the advantages offered by servers in your network.

Methods of Starting Servers

There are several methods used to create servers. The method used affects

- The attributes of the server
- Whether the server runs in the foreground (with an input window and transcript pad) or in the background (without an input window or transcript pad)
- Whether the server process runs on a local or a remote node

In most cases, use the Display Manager (DM) process creation commands, **cp** (CREATE PROCESS), **cpo** (CREATE PROCESS ONLY), or **cps** (CREATE PROCESS SERVER), to run server processes on a user's node. Issue these commands from the DM input window or place them in a start-up file. The command and start-up file used dictate the attributes of the server. Commands issued from the DM input window start server processes immediately. Commands placed in start-up files execute when the file executes. Chapter 2 describes both DOMAIN and DOMAIN/IX start-up files and when they execute.

If the Server Process Manager (**spm**) is running on a node, you can create processes on the node from another location. These special-purpose servers are always running on DOMAIN Server Processors (DSPs), so that you can create other servers or processes on the DSP from a remote node.

Use the shell command **crp** (CREATE REMOTE PROCESS) to create processes on any node from a remote node. The **crp** command options specify the attributes of the process created. These options provide remote processes with attributes similar to those specified for local processes by **cp**, **cpo**, or **cps**.

Refer to the *DOMAIN System Command Reference* for complete information on the DM commands **cp**, **cpo**, **cps**, and the shell command **crp**.

Use the **-n** *server_name* option with server creation commands so that you'll be able to identify a server process in the list output by the **pst** (PROCESS STATUS) shell command. The **pst** command lists the processes running on a node. If you do not give the server a name, and it does not name itself by default, the operating system identifies it as "Process number." We recommend that you use names for servers (either the defaults that some servers provide or a name of your choice) to avoid confusion.

Some servers take options and arguments in the DM command line. Generally, when server processes start from the DM command line, shell command line features are not used. However, some server programs that start in the DM command line do use command line features. In these cases, the DM passes command line standard options to the server program. See the individual server descriptions for information about servers that use command line features from the DM command line.

Server processes can use configuration files for their options and arguments. Some server processes expect configuration files, and these are explained under individual server descriptions. You can create a configuration file for any server that takes options. The file executes when you initiate the server process. The configuration file is called a *names* file. The following syntax:

```
cps /com/sh -c 'server_name configuration_filename'
```

causes the **cps** command to create a shell process that takes the server name and configuration filename as arguments. The shell process controls the server, but the server has the attributes of any process started with the DM command **cps**. As with any shell process, a server process created with this syntax can use command line features.

When you use the DM **cp** command to start a server, you can specify the process window size. Precede the DM **cp** command with the coordinates for the upper left and lower right corners of the window, as follows:

```
(x1,y1)dr;(x2,y2)cp
```

The first set of coordinates is for the upper left corner; the second set is for the lower right corner. See the *DOMAIN System Command Reference* for more information about specifying window sizes.

Attributes of Servers

Table 6–1 summarizes the information from the *DOMAIN System Command Reference* on the **cp**, **cpo**, and **cps** process creation commands.

Table 6–1. Process Start–Up Attributes

Server Start–Up Method	Does process run in foreground or background?	SID of Process ("id" = node ID)	Process on local or remote node?
cp DM command	Foreground, ends on logout.	log–in account SID	Local
cpo DM command	Background, ends on logout.	log–in account SID	Local
cps DM command	Background, runs after logout (except for the **siologin** server).	user.server.none.id	Local
'node_data/startup[suffix]	Background. Commands with **cpo** or **cps** runs whether or not anybody is logged in. **Cp** not used.	user.server.none.id	Local
'node_data/startup.spm DSP [DSP only]	Background. Runs whether or not anybody is logged in. **Cp** not used.	user.server.none.id	Local

Table 6-1. Process Start-Up Attributes (Cont.)

Server Start-Up Method	Does process run in foreground or background?	SID of Process ("id" = node ID)	Process on local or remote node?
crp -cpo	Background, ends on logout.	log-in account SID	Remote
crp -cps	Background, runs after logout.	user.server.none.id	Remote
startup_login[suffix]	Commands with **cp** run in foreground, commands with **cpo** run in background. Both end after logout.	log-in account SID	Local
home_directory/user_data /startup_dm[suffix]	Commands with **cp** run in foreground, commands with **cpo** run in background. Both end after logout.	log-in account SID	Local
shell command line	Foreground, ends after logout.	log-in account SID	Local
shell command line "&" option	Background, ends after logout.	log-in account SID	Local

Maintaining Existing Servers

Check on the status of network server processes with the shell command **pst**. The **pst** command lists all processes running on a node. The **-n** *node* 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** output does not show the server process, use the directions in this chapter to restart the server.

If **pst** does not list a server that you started from the DM input window, or by means of a **crp** command or start-up file, the server might not have started properly. ACLs that do not allow access to the server process SID or to the *'node_data* directory can prevent servers from starting. You should not experience this problem if you use the ACLs we provide in the ACL templates. However, if you change ACL entries and then experience a server start-up problem, check the ACLs for

- The server program itself. Any person or process starting the program must have Execute rights to the server.
- The *'node_data* directory. Any person or process starting any server must have Add, Read, and Write rights to this directory. The SIO line servers have additional requirements. Refer directly to the description of SIO server processes for these requirements.

Pst sometimes lists the server process as running even though the server has stopped. When this happens, you must stop the server process explicitly with the **sigp** (SIGNAL PROCESS) command. Then restart it from the DM window. Refer to the *DOMAIN System Command Reference* for explanations about the **sigp** command. To stop a server process running on a remote node, you must use the **crp** command, log in to the remote node, and then use **sigp** to stop the process.

Alarm Server

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

```
$ help alarm_server
```

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 the *~user_data/startup_dm.*[suffix] file.

The Alarm Server can report on

- Potential disk overflow. You are notified when the disk containing your entry directory starts to run out of free space.
- Severe network problems. You are notified whenever there is a new network hardware failure message. The hardware failure message is described in the documentation for the **netstat** command.
- Observer reports from **netmain_srvr**. You are notified whenever one of the observers in the **netmain_srvr** program makes a report. This option gives immediate notice when the network has problems detectable only by **netmain_srvr**.
- Brief messages from other users. You can use the **send_alarm** program to direct short messages to other users or nodes. Each condition is checked once every four minutes, or at some other interval set by the **-period** option. Alarms may not be posted every time the condition is checked. See the description of each alarm to find out what that alarm's scheduling policy is.

By default, an alarm is accompanied by a distinctive tone pattern. You may, if you like, 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. Using <CTRL> Q before the message appears kills the Alarm Server.

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

Certain optional software packages also make use of the Alarm Server. See the release notes and documentation 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:

```
cpo /sys/alarm/alarm_server -[options]<RETURN>
```

The server process begins immediately and ends at logout.

From a *~user_data/startup_dm.*[suffix] file, enter

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

The server process begins when the named user logs in and ends when that user logs out.

Configuration Files

To execute Alarm Server options from a configuration file, create a file using the format illustrated in Figure 6-1.

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

Figure 6-1. Sample Alarm Server Configuration File

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

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

uses the standard command line option, the asterisk (*), to point to the executable options configuration file. The process names itself "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** alarms 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 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 no pathname is specified, the file *~user_data/alarm_server.netmain_srvr_list* is used. The files should contain lists of node names or hexadecimal node ID numbers separated by spaces or on different lines. 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 in **node_spec** with a hexadecimal 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.

-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. This option is not valid if you choose **-nobell** to suppress all audible alarms.

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 between the horizontal coordinates of each alarm window and **dy** is the difference between the vertical coordinates (in pixels). 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. 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 first alarm window and the size of the alarm windows.

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

Examples

The following example posts nonaudible alarms when the disk is 98 percent full (2 percent free space).

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

The following example posts alarms when this node's 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." Set the alarm window position in the upper left-hand corner of the monitor's screen.

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

Next is a very useful form of the command. It lets you create an options file (in this case *~user_data/opts*) and use that file to control the Alarm Server. The Alarm Server's process is named by the **-n** option. In this case, the process is called "alarms."

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

Special Considerations

Occasionally, the Alarm Server prints a warning message about a file called *.../alarm_server.msg_mbx* as it starts up. These messages can come from several sources. Often the problem is caused by incorrect ACLs on the mailbox (see **mbx_helper**) used by the server process. The Alarm Server can usually change the ACLs on its message mailboxes automatically, but it will sometimes need your help.

The ACLS on two files must allow the **mbx_helper** program Read and Write 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 on using Alarm Server with **netmain_srvr** is given in the section on the **netmain_srvr** process.

Additional information is available for using the Alarm Server with certain optional software packages. For example, the DOMAIN Software Engineering Environment (DSEE) is an optional software package that uses the Alarm Server. See the DSEE manuals for additional information if DSEE is installed on your system. The DOMAIN Professional Support Service (DPSS) also uses the Alarm Server. See the DPSS manuals for additional information if you have DPSS in your network.

mbx_helper – The Mailbox Server

The **mbx_helper** (*/sys/mbx/mbx_helper*) assists processes on different nodes that communicate using mailboxes. This server must run on any node that sends or receives mailbox communications across the network. Beginning at SR9.5, the **mbx_helper** process is automatically started by any Apollo servers or programs that need it to operate correctly. For example, 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.

Online help is available by typing

```
$ help mbx_helper
```

For further information about starting **mbx_helper** from programs, see *Programming With System Calls For Interprocess Communication*. For purposes of providing network services to a node, enable **mbx_helper** by removing the pound (#) sign from the appropriate *'node_data/startup[suffix]* script on that node.

Special Considerations

In a secure network, a mailbox receives its ACL 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 ACLs you do 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 System Calls For Interprocess Communication* for more information. Note that you cannot change a mailbox's ACL while the mailbox is in use.

netmain_srvr – The Network Maintenance Server

The **netmain_srvr** (*/sys/net/netmain_srvr*) collects data necessary for maintaining the network. Use the data for monitoring network performance and isolating potential problems. Use the Netmain Interactive Tool, described in *Managing Your DOMAIN Network*, 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 network. 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 certain unusual conditions. If an observer detects such a 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/net_log* directory by default.

If **netmain_srvr** does not start properly, a record of the failure appears in the file *'node_data/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. **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 *DOMAIN System Command Reference* provides information on **netmain_chklog** and **netmain_note**.

The **netmain_srvr** process should run as a background process from *'node_data/startup.[suffix]* 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 DSPs. Whenever possible, run a monitor in each loop so that data can be collected even if the loop is switched out of the main network. 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
```

Data Collected by netmain_srvr Probes and Observers

This section describes the data collected by the probes and observers run by **netmain_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

Records performance statistics about each node's CPU usage and writes them to *'node_data/net_log/net_log.yy.mm.dd* or a file you specify.

Aegis CPU time: Shows the time the operating system on each node spends on internal needs, and on servicing low-level 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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

DISK_ERRS – Disk and storage module errors

Records cumulative information about disk and storage module performance and errors on all nodes and writes them to *'node_data/net_log/net_log.yy.mm.dd* or a file you specify.

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) errrors 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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

ERR_COUNTS – Network error counts (normal traffic)

Records cumulative network performance statistics and error counts on all nodes and writes them to *'node_data/net_log/net_log.yy.mm.dd* or a file you specify.

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 perfor mance 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 **netstat** 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 exepected 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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

EST_TOPOLOGY – 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

Records every change in the hardware failure message reported by the **netstat** 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.

Hardware failure messages are generated by AEGIS. They appear when there is no network traffic, and AEGIS is completely unable to send network messages.

Default Probe Interval Time: 0:01:00

Default Probe Skip Distance: Not Applicable

MEMORY – Records counts of memory errors on nodes in the network

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

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.

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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

PAGING - Diskless/partner information

Records information about diskless nodes and their paging partners.

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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

SWD_10_MSGS – Software Diagnostic Messages (10)

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.

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 Software Diagnostic 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 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 time 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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

SWD_100_MSGS – Software Diagnostic Messages (100)

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

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.

Default Probe Interval Time: 3:00:00

Default Probe Skip Distance: 1

TOPOLOGY – Total node list estimate

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.

Default Probe Interval Time: 1:00:00

Default Probe Skip Distance: Not Applicable

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

Starting and Stopping netmain_srvr

From the DM command line, type

```
cps /sys/net/netmain_srvr options_filename [-options] <RETURN>
```

The process begins immediately and continues after log out.

From the *'node_data/startup.[suffix]* file, type

```
cps /sys/net/netmain_srvr configuration_filename [-options]
```

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 the shell command **sigp** as shown:

```
$ sigp netmain_srvr -q
```

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 **-cmdf** option. If you specify **-cmdf** *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).

-append 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.

-cmdf *[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, 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 filename 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.

netman – The Diskless Node Server

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 files, use a disked node's operating system to function.

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 Chapter 2 for more information on node start-up directories.

The **netman** server runs as a background process from the *'node_data/startup.[suffix]* file. The command line that executes **netman** is in the default */sys/dm/startup_template* script that arrives 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 section.

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.

Online help is available by typing

```
$ help diskless
```

Starting and Stopping netman

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

```
cps /sys/net/netman<RETURN>
```

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 *'node_data/startup.*[suffix] 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 described above, the process stops running if the node is shut down intentionally or because the system crashes. Restart the process if it stops running by rebooting the node, or from the DM command line.

Special Considerations

Netman allows any disked node to continue functioning even if its disk becomes nonfunctional. For this temporary procedure, get the node ID of any node running **netman**. (You may wish to start the **netman** process on a disked node in the same loop as the now diskless node.) Use the DM command **shut** to bring the node that you want to boot diskless down to the Mnemonic Debugger. Then type, at the MD prompt

```
> RE<RETURN>
> DI N node_id<RETURN> {node_id = node running netman}
> EX AEGIS<RETURN>  Network Partner ID nnnnn
```

Note that in this procedure **netman** bypasses the */sys/net/diskless_list*. It isn't necessary to edit the list to get the node back in the network. When you use this procedure, **netman** will create a *node_data/startup*.[suffix] file for the node with disk problems.

ns_helper – The Naming Server Helper

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. A later section in this chapter describes the procedures you follow to implement **ns_helper** in your network. **Ns_helper** performs most of its operations automatically. **Edns**, 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 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. **Ns_helper** propagates changes in the database of any replica to all other replicas 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 *'node_data/startup*.[suffix] file so that it will continue after logout. **Ns_helper** names itself "ns_helper" by default, so you need not specify the **-n** option to the process creation command.

Starting and Stopping ns_helper

To start **ns_helper** from the DM command line, enter the following. The server process begins immediately and persists after logout.

```
cps /sys/ns/ns_helper<RETURN>
```

To start it from the *'node_data/startup.*[suffix] file, uncomment the line

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

The server process begins when the node comes online and continues after logout.

With both start-up methods described above, **ns_helper** will continue running until it is intentionally stopped with the shell command **sigp**, as shown:

```
$ sigp ns_helper -q
```

If **ns_helper** stops running, you can restart it from the DM command line or, if necessary, by rebooting the node.

Special Considerations

When you run more than one **ns_helper** process in your network, place a server inside and outside of loops which are switched out regularly.

prsvr – The Print Server

The print server process, **prsvr** (*/com/prsvr*), manages files submitted to the print queue with the shell command **prf** (PRINT FILE) or the menu-driven print command **prfd** (PRINT FILE DISPLAY). Create a print server by executing the **prsvr** command and specifying the configuration file for the printer type. This command should execute only when you start or restart the node connected to the printer. Do not execute the command at other nodes; this will cause print files to be lost.

The print server supports the following printers:

1) NEC Spinwriter™ 7715, connected via an SIO line.

2) Printronix™ 300/600 LPM printer, connected to the PBU.

3) VERSATEC™ V-80 or V-82 series printer-plotter, connected to the PBU via an IKON 10071 or 10085.

4) GE series 3000 printer, connected via an SIO line.

5) IMAGEN™ LBP10 and IMAGEN CX printers, connected via one of the following: SIO line; VERSATEC IKON 10071 or 10085, or user-supplied MULTIBUS™ controller.

6) DOMAIN/LASER-26, a 300-dots-per-inch, 26-pages-per-minute laser printer, connected via an SIO line or IKON 10085 MULTIBUS controller.

7) APPLE LaserWriter™, a 300-dots-per-inch, 8-pages-per-minute laser printer, connected via an SIO line.

Typically, print servers are started as background processes from the *'node_data/startup[suffix]* file. 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 node that hosts the print server and the attributes you want the **prsvr** process to have.

Online help is available by typing

```
$ help prsrvr
```

or for details about device configuration files,

```
$ help prsrvr config
```

For details about writing your own device drivers, type

```
$ help prsrvr device_drivers
```

Starting prsvr

From the DM command line, on the local node, start **prsvr** by entering

```
cps /com/sh -c '/com/prsvr' configuration_filename -n process_name
```

The server process begins immediately and continues after log out.

From the shell command line, to start a prsvr process at the time you want to use the printer, type

```
$ prsvr [configuration_filename] [&] [options]<RETURN>
```

Configuration file name (optional)
: You may specify a name; if you don't, **prsvr** looks for *printer_config.data*.

& (optional)
: This character creates a separate shell process in which to run the print server. This process is created without the normal pads or windows, thus running invisibly in the background. When the print server is started in this fashion, it stops automatically at logout. To stop this background process before logout, use the **sigp** command.

-n (optional)
: If you do not use the **-n** option, the print server process is called *print_server.printername* by default.

The *'node_data/startup*[suffix] or *'node_data/startup.spm* file starts the process if you remove the comment character (#) from the following line in those files:

```
# cps /com/sh -n print_server
```

You can also edit this line to include the name of the print server configuration file. Place the configuration file in any directory; however, it usually is convenient to place it in */sys/print*. For example,

```
cps /com/sh  /sys/print/spinwriter -n print_server
```

Unless directed elsewhere, **prsvr**'s default search for the configuration file starts in the current working directory. **Prsvr** follows the operating system's usual search rules (defined in the *DOMAIN System User's Guide*.)

The server process begins when the node comes online and continues after logout.

Starting prsvr from a Remote Node

From a remote node, the shell command **crp** starts the print server on nodes running **spm**. Use the **crp** command in one of three ways:

The first method creates a window to the process on the node and prompts you to log in. Processes started with this invocation of **crp** run in the foreground and end at logout. That is, they have the same attributes as a process started with the **cp** command from the DM input window (see Table 6-1). For example,

```
crp /com/prsvr config_file -n print_server -on node
```

The second method runs the process in the background (there is no window to the process), and the process ends at logout. These are the attributes of processes started using the **cpo** command from the DM input window (see Table 6-1).

```
crp /com/prsvr config_file -n print_server -on node -cpo
```

The third method starts the process in the background, and the process continues after logout. These are the attributes of processes started using the **cps** command in the DM input window (see Table 6-1). A process started with this method can be stopped with the shell command **sigp**. It must be restarted if the node stops running the process for any reason. For example,

```
crp /com/prsvr config_file -n print_server -on node -cps
```

Stopping prsvr

With the DM command and the *'node_data* start-up methods described above, **prsvr** will continue running until it is intentionally stopped with the shell command **sigp**, as shown:

```
$ sigp prsvr -q
```

With both startup methods, the process stops running if the node is shut down intentionally or because the system crashes. If it stops running, restart the process from the DM command line or by rebooting the node.

There is a special application for

```
$ sigp prsvr -s
```

which tells **prsvr** to stop printing a file that is currently printing and delete the filename from the print queue. If a file is not currently printing, the **-s** option reverts to its usual meaning, which is stop the process entirely.

> NOTE: The **sigp -s** command should be used in this manner only with line printers, for example a Spinwriter. Do not use it with printers that have intelligent print controllers, such as laser printers, or machines where you can delete waiting jobs.

Configuration Files

To start printers other than the default printer (p), or to use **prsvr** options, create a printer configuration file. Figure 6-2 shows example configuration files.

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>
```

Figure 6-2. Sample Print Configuration Files

You can specify the name of the configuration file in the start-up file, for example,

```
cps /com/sh `/sys/print/ptx_config` -n print_server
```

If you use the shell command **crp** to start the print server, type

```
$ crp '/com/prsvr /sys/print/ptx_config' -n ps -on node -cps<RETURN>
```

In each example above, the configuration file is named *ptx_config* (for the Printronix printer). Give a different name to each printer 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).

Print Configuration File Options and Arguments

Some of the options and arguments below can now 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.

bottom_margin *[n]* Number of lines to skip at the bottom of the page. The **prsvr** uses this option only if the **prf** or **prfd** command does not contain a margin specification. The default value is four (4) lines.

collate_copies*[on|off]*
Enables page collation for multiple copies if ON.

cpi *[n]* The default characters per inch (the pitch) printed by the current printer. May be overridden by user options to the **prf** command. Use this option if you use a non-default pitch for a printer. The following list shows the defaults for each of the printers we support. See also **print_width**.

```
IMAGEN   12 cpi
SPIN     12 cpi
PRINTR   10 cpi
VERS     12 cpi
GE       12 cpi
```

ddf_name *[pathname]*

Specifies a device descriptor file for a MULTIBUS controller. This option is useful for driving a printer with MULTIBUS card. Printers we supply that may use this option are the IMAGEN and VERSATEC printers. The default name is */dev/versatec*. See the *GPI/O User's Guide* for more information on this option.

device [spinwriter | printronix | versatec | ge | imagen | user1 | user2 | user3 | user4]

Specifies the DOMAIN-supported printer types or user-supplied device drivers that print files. See reference material on the **usern** argument. The default device is the Spinwriter.

file_banners [off|first|last]

Causes a banner page to precede each file. May be overridden by user options to the **prf** command. **Off** means no banner page; **first** means the banner page is printed before the job and **last**, after the job.

form_feeds *[n]* The number of pages to form feed between jobs. The default is 1 form feed (paper advances from current page to top of next page.).

interface [serial | versatec | multibus]

Indicates 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.

lpi *[n]* 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. The following list shows the defaults for each of the printers we support. See also **print_length.**

```
IMAGEN    6 lpi
SPIN      6 lpi
PRINTR    6 lpi
VERS      6 lpi
GE        6 lpi
```

logo [string|none] Specifies a character string to be printed on the banner page. The default is none (no logo is printed).

model_number *[xxx]* Currently, this option applies to printers with multiple model numbers such as

```
IMAGEN    8/300       for the CX
LBP-10      for the LBP-10
VERSATEC V8236
         V8224
         V8244
         V8272
         V80
```

pageno_column *[n]*

Column in which the page number is printed in the header. May be overridden by user options to the **prf** command. The default column is 90.

page_headers [on|off]

If **on**, print 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. The default is **on**.

page_length *[n]* Length, in lines, of the page. This is the default option if **print_length** is not specified. The default length is 66 lines.

page_width *[n]* Width, in characters, of page. If the input line length exceeds the specified page width, the excess characters are truncated and a warning message appears, listing the number of truncated lines. This is the default option if **print_width** is not specified. The default width is 132 spaces.

page_reversal [on|off]
Order of pages printed. If **on**, the pages of the file are printed last page first; if **off**, first page is printed first. (Supports PostScript interpreter.)

plot_mode [on|off] Specifies whether the device will accept plot files. The default mode is **off.**

print_length *[n]* Specifies in inches the length of paper that 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 a paper size of 8.5 by 11 inches but can only print on an area size 8.0 by 10.8 inches. Use this command instead of **page_length** to specify page format. See also the example configuration files and the **lpi** option.

print_width *[n]* Specifies in inches the width of paper that 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. See also the **cpi** option.

printer_name *[string]*
Specifies printer name used in the **-pr** option of the **prf** command; useful when several printers are attached to a single node. The default string is p.

resolution *[n]* Specifies 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 defaults are as follows: CX: 300; LBP10: 240; and GE: 72 dots per inch.

sio_line *[n]* Specifies the SIO line to which a printer is attached; not meaningful for Printronix or VERSATEC printers. The default is line 1.

speed *[n]* Specifies baud rate for the SIO line; not meaningful for Printronix or VERSATEC printers.

top_margin *[n]* Number of lines to skip at the top of the page. **Prsvr** uses this option only if the **prf** or **prfd** command does not contain a margin specification.

The device user*n* Option – User-Written Device Drivers

The **prsvr** program can support as many as four customer-supplied device drivers on systems running Software Version SR4.1 or later. This support is in addition to that supplied for the NEC Spinwriter, Printronix, VERSATEC, IMAGEN, and GE 3000 printers.

If you write your own device driver to run a printer you supply, it must include the six procedures listed below. Use the names shown. Replace *n* with the device number that corresponds to the number in the **device user***n* option in the configuration file.

The declarations of the following calls and data types used may be found in */sys/ins/prsvr.ins.pas*.

- USERn_INIT (sio_line, sio_speed). The **prsvr** process calls this procedure once to initialize the printer. The procedure typically initializes the internal state of the driver. If you are using an SIO line, SIO_$CONTROL should be called to set up the characteristics of the line. The SIO_SPEED parameter, declared in */sys/ins/sio.ins.pas* or */sys/ins/sio.ins.ftn*, indicates the line speed of the

selected SIO line (i.e., one of SIO_$50, SIO_$75, SIO_$110, . . . SIO_$19200). A stream call should be made to open the SIO line.

- USERn_WRITE (string, length). The **prsvr** process calls this procedure to pass the specified string to the device. The string may be ASCII text and include 0 or more newline characters, or it may be plot data. It is the driver's function to correctly interpret this data.

- USERn_SET_MODE (mode, value). The **prsvr** process calls this procedure to pass the device driver a pointer to an attribute block. The attribute block contains information that the user may have specified at the time the file was enqueued. Refer to */sys/ins/prsvr.ins.pas* for the contents of the attribute block named SERVER_DB_T.

- USERn_RETURN_INFO (query, info). The **prsvr** process issues this call to determine the capabilities of the device from the user-supplied device driver. The data is passed to the server as a pointer to a record structure called DRIVER_DB_T, defined in */sys/ins/prsvr.ins.pas*.

- USERn_FLUSH. The **prsvr** process calls this procedure to flush the output buffer for the device. Ignore this call if the device driver does not buffer data for the device.

- USERn_CLOSE. The **prsvr** process calls this procedure to close the device.

After you write the device driver, bind it to the print server as follows:

```
$ bind -b prsvr.user /com/prsvr usern.bin<RETURN>
```

The binder will notify you of undefined globals. For device driver *usern.bin*, ignore all undefined globals except those that refer to *usern*.

The Interface Multibus Option – Interfacing the IMAGEN Printer

The **interface multibus** option for the laser printer allows you to use MULTIBUS controllers other than the VERSATEC IKON 10071 and 10085 boards. To use this option, write a program module containing the following entry points:

```
multibus_$init(sio_line,sio_speed) performs any required initialization.

multibus_$write(buf,buflen) passes a buffer (buf) of type univ pr_$buf_t and
length (buflen) to the controller.
```

These entry points should be specified in the Device Descriptor File (ddf) for the controller. The actual binding takes place at run time. Be sure to specify the **ddf_name** in the **prsvr** configuration file **ddf_name** option.

Special Considerations

Note how the default printer_name (p) is used in the sample Printronix configuration file in Figure 6–2. Users can send files to this printer, without using the **-pr** option in the shell command **prf**. In the sample IMAGEN file, also in Figure 6–2, the **printer_name** option is specified as **cx**. Users must specify **-pr cx** when they send files to this printer.

You can configure printers so that, if one is busy, another one will automatically start printing. You do this by specifying configuration files with the same **printer_name** option.

You must create a link, on every node in the network, to the */sys/print* directory of nodes connected to printers. You can distribute printer resources by linking some users to one */sys/print* directory and linking other users to a different */sys/print*. Users always can specify different */sys/print* directories with the **-s** option of the **prf** command.

The new print configuration file options, **cpi, lpi, print_length,** and **print_width** are meant to be used together to give **prsvr** information on page format. Note that **cpi** and **lpi** would be used only if you are using non-default options on your printer. Usually **print_length** and **print_width** will be sufficient to set up a page format. Use these options in preference to the options **page_length** and **page_width.**

Related Information

Refer to the documentation for the GPI/O package for further details on using the MULTIBUS and writing device drivers.

SIO – Serial I/O Line Servers

The SIO line servers allow you to connect a "dumb" terminal to a DOMAIN workstation, either directly or via modem and telephone lines from another location. These servers manage a DOMAIN workstation's asynchronous I/O (SIO) lines and the process of logging on to the connected terminal.

The SIO line server processes are

siomonit [SIO PROCESS MONITOR] (*/sys/siologin*)
Typically invoked as a background server process from the *'node_data/startup[suffix]* file.

siologin [SIO LINE LOGIN] (*/sys/siologin*)
Invoked by **siomonit.** It must be a manager within the login protected subsystem. **Siologin** is the process that directly manages user login.

Below, we describe the procedure used to connect terminals to DOMAIN 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 reference sections that follow this one.

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

1. 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

2. Change the SIO line configuration to conform to the modem or terminal configuration parameters. This may be done in a shell command file run by **siologin** when it starts. Alternately, change the terminal or modem configuration parameters according to the manufacturer's instructions to conform to the SIO line configuration.

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 DOMAIN workstation. Connect the modem to the telephone. When you are ready to login, signal the SIO line by typing <RETURN> on a locally connected terminal. From a remote terminal, dial the number of the phone line connected to the node's modem.

When you are finished using a local or remote terminal, type <CTRL> Z (or the character you have defined as the EOT 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.

siologin – The SIO Line Log-in Server

The **siologin** process uses the following format.

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

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*. **Dev_name**, 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/sio1*,

```
/com/sh 'node_data/startup_sio.sh 1
```

Include **tctl** commands here to configure the line or include the line

```
$ ulkob ^1 -f<RETURN>
```

to ensure that the SIO line is not locked through some previous failure.

Online help is available by typing

$ help siologin

or

$ help protection protected_subsystems

siologin Options and Arguments

The **siologin** process accepts the following arguments.

dev_name (required)
: The SIO device descriptor pathname, in the form */dev/siox*, 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.

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 remote lines,

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.) **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 6-3 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 6-3. Sample 'node_data/siologin_log

Note that to operate, the **siologin** server must have manager status in the login protected subsystem (see the *DOMAIN System User's Guide* for more information on the subsystem.) The DOMAIN 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<RETURN>
```

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:

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

the **siologin** process has lost its manager status. To reassign manager status to **siologin**, you must log-in with a *sys_admin* account and type

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

siomonit – The SIO Process Monitor

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 describe 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:

```
cps /sys/siologin/siomonit siomonit_filename<RETURN>
```

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 the *'node_data/startup.*[suffix] file, uncomment the line that reads as follows:

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

Be sure this line appears after any lines in the startup file which 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 wish 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 <RETURN>
```

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 signals **siomonit** to restart the **siologin** process. This argument should be the first one given.

siologin_arg_list

The list of **siologin** arguments and options described in the section on **siologin**. Arguments are passed to **siologin** unvalidated; however, the first argument must be */dev/sion*. **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 6-4 shows a sample *'node_data/siomonit_file*. Note that comments may be included by placing the "#" sign at the beginning of a line.

```
# Sample `node_data/siomonit_file
#
#  Watch sio lines 1 and 2 and keep them available for siologin::
#     line 1 is a dial up line:
#
-repeat /dev/sio1 -dialin -n siologin1 /com/sh -f -c user_data/startup_sh
#
#     line 2 is a local connection:
-repeat /dev/sio2   -n siologin2_local
#
# up to three siologin arg lists like the ones above may be included, one
#  for each SIO line (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 6-4. Sample 'node_data/siomonit_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 6-5 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/sio1 -n siologin1
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/sio1 -n siologin1
Tuesday, February 19, 1985  9:18:02
 * Couldn't open command input file `node_data/siomonit_file. Status 1010015 *
```

Figure 6-5. Sample 'node_data/siomonit_log

Often, the failure of an **siologin** process to stay alive can be caused by a locked SIO line (*/dev/siox*). For example, when a user on a locally connected terminal does not end the session with <CTRL> Z, the SIO line to that terminal remains locked. To free up the line, use **sigp -q** to prod **siomonit** into trying again. After you free the line, you may want to include **ulkob ^1 -f** in your *'node_data/startup_sio.sh* 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.

spm – 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. **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 do the following things:

- Create processes from a remote node by using the shell command **crp** with options similar to the Display Manager's **cpo** and **cps** commands. The process you create may run another server, such as **prsvr** or **netman**. The process can also run a shell program.
- Log in to the node for debugging purposes or to maintain servers. For example, you might want to use the shell command **sigp** to stop a server process running on the node.

Starting and Stopping spm

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

```
cps /sys/spm/spm -n server_process_manager
```

The process begins immediately and continues after logout.

To invoke it from the *'node_data/startup*[suffix] file, uncomment the following line in the file:

```
# cps /sys/spm/spm -n server_process_manager
```

The processes 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
```

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. When the **spm** runs in place of the DM, it waits on the eventcount file *'node_data/spmshut_ec*. The **shutspm** command advances this eventcount, causing the **spm** to perform an orderly shutdown of the node.

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

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.%       pgndwrx
%.%.%               dr
```

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.

To prevent **spm** from responding to the **shutspm** command, add the following line to the *'node_data/startup.spm* file:

```
no_shutspm
```

Tablet Server

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 *'node_data/startup[suffix]* file. The server process sends a control character to the tablet bit pad and to the **-bp_enable** option of the shell command **tctl**. An operating system program then provides the actual bit-pad support. The process started in the *'node_data/startup[suffix]* 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 *'node_data/startup*[suffix], or *'node_data/startup.spm* files:

```
# 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
```

Chapter 7

Collecting Information about the Network

You need information to guide you when you allocate existing network resources or troubleshoot problems. Your service representative often needs detailed information about the network to diagnose and repair problems.

This chapter describes the information you need to collect to manage the network effectively. It reviews sources of network data and describes how to generate the information. The next chapter describes how to use the information to manage the network.

The key issues in managing the network are

- Allocating network resources efficiently
- Ensuring the integrity of the physical media

Allocating network resources means distributing them so that they will be used well. Network resources include tangible items such as disks, CPUs, server nodes, and printers, and information resources such as network databases, registries, master root directories, and software libraries. Ideally, resources should be used efficiently, and equitably; that is, neither under- nor overutilized, and placed where they are most needed.

Ensuring the integrity of the physical media means maintaining the nodes, transmission media, and switches so that the network is reliable and available at all times. It also means that you are able to quickly isolate broken cables and nonfunctioning nodes or disks, and remove them from the network until they can be repaired.

We assume that you are familiar with **netmain_srvr** and the Netmain Interactive Tool; **netmain_srvr** is described in Chapter 6, and the Netmain Interactive Tool in Chapter 10 of this book.

When you used the book *Planning DOMAIN Networks and Internets* to arrange your site, you began to collect some of the information that you need. The first section of this chapter reviews this information. The rest of the chapter reviews other sources of information you need to manage your network.

The Network Site – Topography

You should know the physical location of all the nodes in the network, the network switch boxes, and the loops that comprise your network. Keep this information in the form of an accurate diagram of your network's current configuration. The network diagram should contain the following information:

- Location of every node
- Layout of every loop
- Length and location of every piece of cable
- Location of every switch and junction point
- Location of the network control room
- Direction of data flow throughout the network
- Name, ID, and type (e.g. DN3000, DSP160) of every node

If you followed the procedures given in the *Planning DOMAIN Networks and Internets* manual, you prepared most of this information prior to installing your network. Review the diagram now to be certain that it is an accurate representation of the installed network **topography**. Keep the diagram up to date as new nodes are added to the network.

In the event of network problems, you or your service representative may need network topography information to locate and repair a problem. If there is a problem with cable located where you cannot see it, you need the diagram to find the cable location. An accurate representation of the network's topography can save time when locating trouble spots.

Use the network topography information when you allocate network resources. Correlate the topographical information with other information discussed below when changing the location of resources, planning for new ones, or enhancing network performance.

The Network Communication Path – Topology

The network **topology** describes the data transmission stream through points (nodes and switches) in the network. Data in a DOMAIN ring flows in one direction around the ring. We say data flows from a point upstream to a point downstream. When you laid out cable at your network site you decided on the direction of data flow. You can describe nodes as upstream or downstream of each other in relation to their position within the network topology.

While network topography is relatively stable over long periods of time, network topology can change rapidly as nodes and loops are switched out of the ring. There are several methods of generating a network topology list. Use each method for a different purpose described in the sections that follow.

The lcnode Command and Network Topology

The **lcnode** command gives the network topology information necessary to perform network troubleshooting as outlined in this book. When you execute the shell command **lcnode** (List Connected Nodes) it lists

- The direction of data flow in the network; that is, the output lists the node from which you issue the command, then all the nodes downstream of that node
- The nodes currently connected and responding to the command
- The paging partners of diskless nodes

The **lcnode** command tells you what nodes are on the network at the time you issue the command. If you execute **lcnode** at the beginning of the work day, it can provide an operational topology list for that day. The **lcnode** command may not operate, or it may produce incomplete results during periods of network problems. For this reason, you should have the topology information on hand before a problem occurs.

You can execute **lcnode** and view its output in the transcript window. However, you may find it more convenient to copy the output to another file and print it so you will have hard copy of the operational topology list.

The netmain_srvr Topology Lists

The **netmain_srvr** and the Netmain Interactive Tool can provide several kinds of topology lists. You need a Master Topology List which includes

- The direction of data flow in the network; **netmain** output lists the monitor from which information is collected, then all the nodes downstream of that node
- All nodes whether or not they are currently connected to the network
- The 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.

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 */sys/net/net_log* directory. The directory contains logs from all **netmain_srvr** probes and observers. Remember that **netmain_srvr** uses non-ASCII files for its work. You can look at the files in */sys/net/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 the 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 List 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 Topo 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. Chapter 10 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 F6 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 Find Monitors and Nodes menu, F2 command Find All Monitors, will generate this list.

Using netmain_srvr for 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 on node and network performance and error conditions. Chapter 6 defines each of the statistics collected by **netmain_srvr**. 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

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.

Use the **netmain_srvr** configuration file options **-observe** and **-sample** to set selected probes to never, so they simply don't run on a monitor. When you start **netmain_srvr** from *'node_data/startup[suffix]*, the configuration file automatically runs with the unique configuration you've selected for the server process on that node.

To configure probes on a monitor with the Netmain Interactive Tool, use the Change Monitor Behavior menu. Select the F2 Alter Probe Timing submenu to deactivate those probes you don't want to run by setting their schedules to never. The probe will remain inactive on this monitor.

Note that the start-up script execution rules, described in Chapter 3, apply to the **netmain_srvr** started from *'node_data*. If you edit a configuration file belonging to an active monitor, the new configuration will not execute until the next node boot unless you stop the process (**sigp -s**) and restart it from the DM command line. If you use the Netmain Interactive Tool to change the active monitor's configuration, its behavior changes but the configuration file is not affected.

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. Chapter 10 discusses probe scheduling guidelines. 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. Alarm Server should execute from */sys/node_data/startup_login[suffix]* for this purpose.

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.

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 that indicate hard breaks in the network cause nodes to report a failure in the network. The chapter on troubleshooting fully describes this 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 7-1 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 7–1. 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
Receive/Transmit ESB Error	Can indicate hard break in network
Receive/Transmit Modem Error	Can indicate hard break in network
Transmit No Return	Can indicate hard break in network
HW_FAIL	Indicates hard break in network

netmain_srvr 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 7–2 shows the **netmain_srvr** node and network performance statistics that can be useful for allocating network resources.

Table 7–2. netmain_srvr Probes Reporting on Network Performance

Probe	Level of Error	
CPU_TIME:	**DISK_ERRS:**	
Aegis CPU	Total Storage Module Activity	
User CPU	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:		
All Items		

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 (see output from the **netstat** command in the troubleshooting chapter), 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 Log a Text String command box in the Change Monitor Behavior menu.
- Outside the Netmain Interactive environment, use the **netmain_note** shell command, as shown below.

Type:

```
$ netmain_note message <RETURN>
```

For example, you might type:

```
$ netmain_note Loop 17 scheduled downtime  <RETURN>
```

To retrieve the notes stored in a log file, use the Printed Output Format in the Netmain Interactive Tool's Analyze Network Data menu.

Node Problem Logs

Keep a log book at each node to record node problems and service. In the node log book, record all known information about a problem. Repeated crashes of the same type may indicate a hardware problem.

In the node log book, include the following information about node crashes:

- Date and time of the crash
- Name of person who restored the node
- The node crash code
- The location of the dump, if taken
- Action taken to restore the node (the SALVOL command, etc.)

The *DOMAIN System Command Reference* and your node's operations guide describe network dumps and crash status codes.

Also include in the node log book the date and time that new software is installed and the service performed by field service representatives.

Chapter 8

Maintaining the Network and Nodes

This chapter provides guidelines for using network topology lists and data gathered by **netmain_srvr** and the Netmain Interactive Tool displays to draw conclusions about your network's performance. It contains methods to detect conditions that are potentially troublesome, and thus is oriented toward problem prevention. This chapter 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

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. Chapter 10 describes how to use the Netmain Interactive Tool to detect resource distribution problems.

This section uses the troubleshooting approach similar to Chapter 9, "Troubleshooting the Network." Chapter 9, however, describes how to make a "broken" network operational as quickly as possible. 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. To increase your understanding of the material in this chapter, 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 Chapter 10 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 horizontal high-density 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 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 section described a quick check to compare diskless partner assignments for nodes of various types. Use the network resource analysis techniques described in the section 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) CRC errors. Set the "top of error scale" to 5%. If a disk or storage module 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.

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 using 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 and 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 section describes how to look at displays of performance statistics to isolate the source of the problems described in the previous sections. 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 8-1 shows an example of such a condition in a density plot.

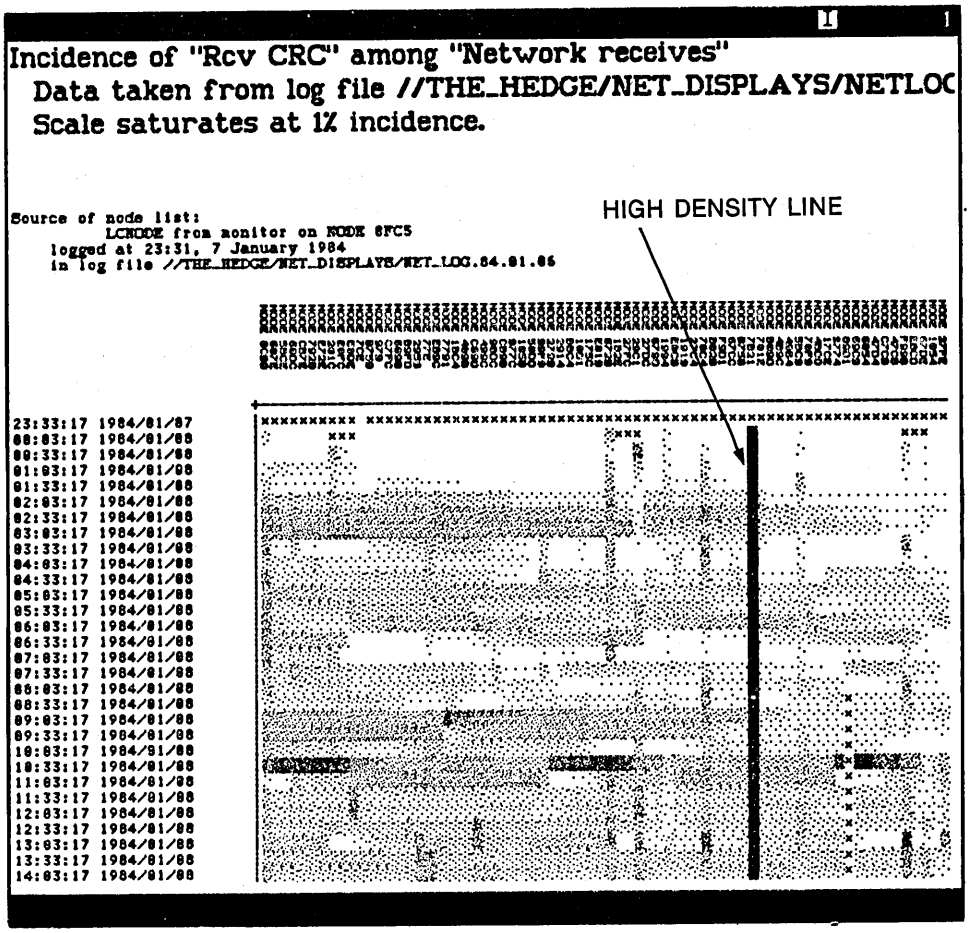


Figure 8-1. High-Density Line Condition

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 problem with a cabling or 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).

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. You can use the "two output pads" technique described earlier to detect sources of problems.

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 8-2 shows an example of high-density fade in a gray scale plot.

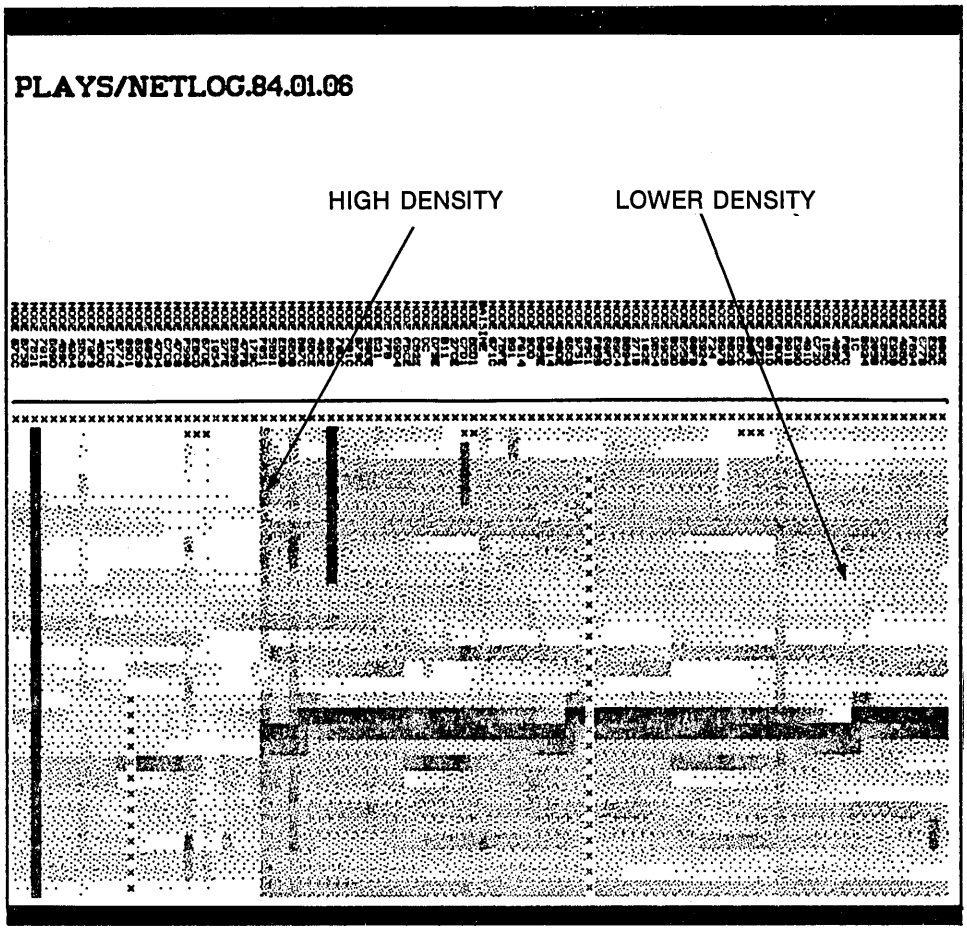


Figure 8-2. High-Density Fade Condition

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 8-3 shows two plots aligned in just this manner.

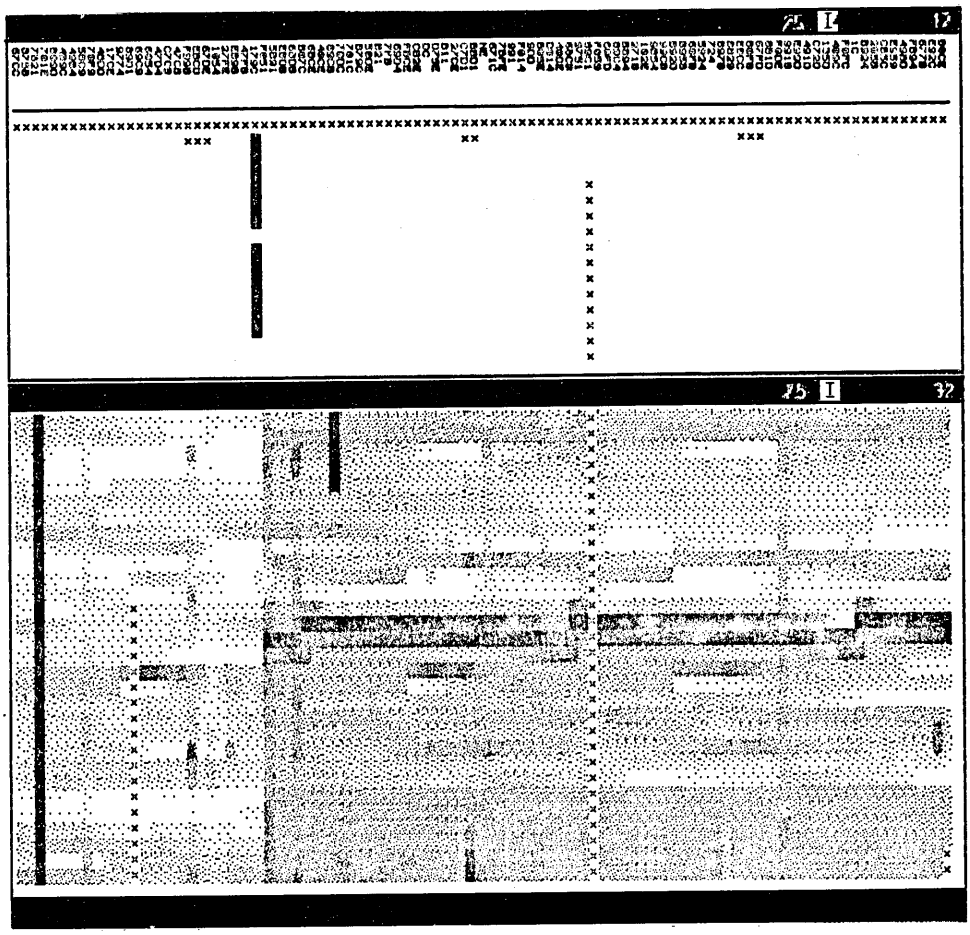


Figure 8-3. Aligned Plots

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.

More Intensive Methods of Locating Network Performance Problems

If you want more information about a node than is provided by the methods described in the last two sections, 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 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.

Routine Maintenance Procedures

Although system administrators are not responsible for major hardware maintenance, there are simple maintenance routines that can ensure the integrity of the network and nodes. Perform the procedures described below to maintain the network and nodes between periods of major service.

General Node Maintenance Procedures

Make and keep a regular maintenance schedule. The maintenance procedures you can perform to maintain the integrity of your network include the following tasks:

- Cleaning floppy disk drive heads monthly
- Vacuuming all printers as needed (see manufacturer's instructions)
- Cleaning storage module filter monthly (see manufacturer's instructions)
- Cleaning display screens as needed

Performing Network Hardware Checks

Problems in coaxial cables and cable connectors cause most network problems. Check all coaxial cables and connections once a month at a time when network traffic is light.

To check the hardware, look at the cable connections for obvious problems and gently touch each set of external cable connections (at the rear of the node). If the network experiences a problem when you move a cable, there may be a problem with that cable or with one of its connectors. Use the directions in the *Installing Coaxial Cable and Accessories for a DOMAIN Token Ring Network* manual to recouple the cable, if necessary.

Do not remove any cable connectors when you make your network checks unless you find an obvious problem. If you do find a cable that requires recoupling, switch out the loop containing the cable before making the repair.

Maintaining Node Integrity

Follow the recommendations in the *DOMAIN Hardware Site Planning Specifications* manual for node clearance and electrical and heat dissipation requirements. Inspect nodes at your site regularly to be certain that heat vents on display monitors are not covered by papers, books, etc. Ensure that users do not attempt to move nodes themselves, and that nodes are plugged into the proper outlets.

Chapter 9

Troubleshooting the Network

This chapter describes procedures for troubleshooting network problems. Network problems might be the cause when any of the following conditions occur:

- Users cannot access remote files.
- The time required to access network resources increases.
- Diskless nodes cannot reach their partners.

The common element among these events is that the software or hardware fails during a remote operation, i.e., when operating across the network. The common causes of network failure are generally either cable or connector faults, or malfunctioning nodes.

The procedures used to isolate the cause of network failure require you to systematically examine successively smaller parts of the network; that is, loops, then nodes, then cables and connections. Gather evidence for the cause of network failure to make an informed decision about the cause of the problem.

The remainder of this chapter describes general techniques for troubleshooting the network and then gives the actual procedures to use.

General Techniques

Carry out troubleshooting procedures under the control of a single person. This means that a single person should decide when and where to perform each of the diagnostic procedures, evaluate the outcome of each test, and decide on the next procedure(s). Use orderly troubleshooting procedures, because you use information collected during the troubleshoot to determine the cause of failure. If several people are performing the procedures, the information you obtain may be confusing or even worthless. To successfully isolate the cause of failure, you must understand both the topology of your network and how the operating system detects network failures.

You need a network topology list to proceed with troubleshooting. If you followed the network maintenance suggestions in Chapters 7 and 8, you have documentation on the topology of your network. The remainder of this chapter describes how to use the information you collect for troubleshooting.

How the Operating System Detects Network Failure

There is software on each node that monitors the node for failures. It reports some conditions as the output of shell commands or of the Netmain Interactive Tool. For example, the operating system reads ring hardware registers and keeps a count of errors. These counts are available from the Netmain's Analyze Network Data menu, F1 command and from the shell command **netstat -l** output. These error counts increase during times of network failure.

The operating system also checks for two conditions that may indicate hard breaks in the network:

- The occurrence of biphase errors indicating a node's failure to lock on to the transmission signal. That is, the node can't get information from the ring.
- The node's failure to perform any network input/output (I/O) activity in the last minute.

When either condition occurs, the node sends a test message across the network. If the test message returns intact, there is no problem; the node simply was not performing any network I/O at that time period. If the test message does not return intact, the node broadcasts a special failure message to other nodes, based on the part of the message that returned. The broadcast says that the node detected a "ring hardware failure," and it describes the failure as one of the following types.

`Node is not receiving clock signals.`

The clock information encoded with the data transmission and the data itself has been lost. This condition generally occurs when the network is physically broken. No packets can get through.

`Node is receiving clock signals, but no data.`

The clock information encoded with the data transmission got through, but the data did not get through correctly. This condition occurs when packets can traverse the ring, but data is lost or corrupted.

If the node is receiving clock information, but the test message does not return to the transmitter, the node reports a "no data" hardware failure. If the node is not receiving anything, not even the clock information gets through. The network is apparently broken, and the node reports a "not receiving clock signals" hardware failure.

A *data loss* hardware failure usually indicates a problem with a node's ring hardware. A *clock loss* hardware failure usually indicates a problem with the cable or connectors between two nodes.

When you troubleshoot a failing network, use the hardware failure message to locate the problem node or cable in the network. The node that reports the failure is usually contiguous to the failure (either upstream or downstream), or is itself the cause of the problem.

In the 60 seconds after the operating system detects a problem, several nodes may attempt to send a test message and then may try to send error reports. This may cause inaccurate information in the **netstat -l** first performed.

When the last failure detected by a node occurred more than a minute ago, the **netstat -l** failure report describes it as the "last failure." The **netstat -l** output continues to report the last failure until a new failure is detected, or until the node is rebooted and the error counters are reset. Figure 9-1 shows examples of failure reports at the end of the **netstat -l** command output.

A clock loss failure occurred less than a minute ago, or a node running early revision software noted a failure of either type less than a minute ago:

```
Ring hardware failure detected by node A59 on 1983/08/31 at 13:16
   Node is not receiving clock signals.
   System configured with 1.0 mb of memory.
   A total of 0 ECCC errors were detected.
```

A data loss failure occurred less than a minute ago:

```
Ring hardware failure detected by node 92B on 1983/09/02 at 15:21
   Node is receiving clock signals but no data.
   System configured with 1.0 mb of memory.
   A total of 0 ECCC errors were detected.
```

The last failure, a data loss failure, occurred more than a minute ago:

```
Last ring hardware failure detected by node FFF on 1983/09/09 at 17:57
  Node is receiving clock signals but no data.
   System configured with 1.0 mb of memory.
   A total of 0 ECCC errors were detected.
```

The node hasn't received a broadcast of a failure report since it was booted:

```
No ring hardware failure report.
  System configured with 1.0 mb of memory.
  A total of 0 ECCC errors were detected.
```

Figure 9-1. Examples of **netstat** *Failure Messages*

Getting Information During Network Failure

The procedures described in this chapter use a "top down" approach to locate the cause of the network failure. First locate the failing loop, using the **netstat** command, and switch that loop out of the network. Next, locate the failing segment of the network in that loop and remove or repair that segment.

Use network and node problem logs, described in Chapter 7, to quickly pinpoint a problem. A similar problem may have occurred in the past at this part of the network.

When there are frequent, but intermittent, network problems, use data gathered by the Netmain Interactive Tool. In particular, refer to error and hardware failure messages. Monitors with low skip distances and small sampling intervals provide the best information.

Use the Netmain Interactive Tool, as described below, when a network problem occurs.

- On a node that runs a monitor, use the Find Monitors and Nodes menu to select the monitor on that node. Use the Change Monitor Behavior menu to determine the "check interval" of the ERR_COUNTS probe. If necessary, reschedule the probe so that it checks each node every few minutes.

- Use the Topo List from Node Total and Print Topology List features to get a topology list, if you don't have one.

- Use the Analyze Network Data menu to get a graphic or printed report of error occurrences gleaned from the monitor itself. If you want to look at the current log file, first close it by using the Change Monitor Behavior menu.

Network Troubleshooting Procedures

This section contains procedures used to isolate the cause of a network failure and to restore the network. The procedures are best used when the network has a multiloop format as described in *Planning DOMAIN Networks and Internets*.

In multiloop networks, a failing loop can be switched out of the network after you determine that it is causing the problem. The rest of the network should become functional again. Most users can continue to work while you isolate, then correct the specific cause of the network failure.

If your network does not have a multiloop format, troubleshooting can take more time. However, if you followed the network documentation guidelines given in Chapter 7 and 8, adequate information for finding the cause of the problem should exist.

The procedures given here are guides for isolating the cause of network failure. Use your knowledge of your own network to make judgments about when and where to apply these procedures. The procedures are for the following tasks:

- Determining if there is a network problem

- Locating the failing loop or loops

- Locating the point of failure

Execute the commands in these procedures from a node with a disk. You cannot use a diskless node if the network has failed. In addition, when you use the Netmain Interactive Tool to assist in network troubleshooting, use the program on a node that runs a monitor or on a node that has a recently closed log file to investigate. You may not be able to access a monitor or log file on a remote node.

Symptoms that appear to be caused by a failing network may have other causes. For example, if users cannot gain access to files on a remote node, the remote node or its loop may be off the network. The files' ACLs may not allow access by the user. The files may be locked. Use Procedure 9–1 to determine if the network has actually failed. If the network has failed, use Procedure 9–2 and Procedure 9–3 to continue the troubleshooting.

PROCEDURE 9-1: Determining if the Problem is a Network Failure

1. Execute **ld** (List Directory) for a disked node that you know is online, in another loop. Specify /com to ensure that the local node executes the command from its own */com* directory, and does not go across the network to find the command. Type the following:

   $ /com/ld //node_name <RETURN>

   If the command output shows a listing of the entry directory, the network is working. If you receive an error message, the network may have failed. Go to Step 2.

2. Execute **netstat -l**. The **netstat** command writes a summary of the network and hard disk activity. The **-l** option provides the long form. Figure 9-1 shows **netstat** command output. Type:

   $ /com/netstat -l <RETURN>

3. Examine the bottom lines of the message. If these lines report that the last ring hardware failure occurred recently, assume that the network has failed. Go to the next procedure. Otherwise, go on to Step 4.

   CAUTION: Do not use **lcnode**, or any Netmain features that perform **lcnode**, to determine if the network is failing. The **lcnode** command polls all nodes in the network. If the network is failing, executing **lcnode** may take a long time and give only partial results.

4. If **netstat** reports no errors, there is probably not a "hard" failure. There may be an intermittent failure or a failure that **netstat** cannot detect. If you still think there may be a network problem, go to a node that has a **netmain_srvr** monitor running or has a log file that you can examine. Then use the following Netmain tools to try to locate the problem:

   - Analyze Network Data menu
   - Error peaks for Transmit modem errors
   - Error density for Transmit modem errors

END OF PROCEDURE 9-1

PROCEDURE 9-2: Locating the Failing Loop or Loops

This procedure describes how to locate a failing loop in the network. The procedure assumes that the network has a multiloop structure with network switches. If the network does not have a multiloop structure, use Procedure 9-3. This procedure instructs you to switch loops out of the network or simply to "switch out" a loop. Do not switch any loops back into the network until instructed to do so.

1. Execute **netstat** as shown, type:

   $ /com/netstat -l -r 5 -s 10 -n node <RETURN>

   [node is the node ID or entry name of some other node in the same loop.] Start another shell process on the same node. In it, execute **netstat** as shown; type:

   $ /com/netstat -l -r 5 <RETURN>

   The **-r** option makes **netstat** report new data every n seconds. In the first report, **netstat** displays the totals for all network statistics it monitors. In subsequent reports, **netstat** displays only the *differences* since the previous report. The **-s** *n* option makes **netstat** send n messages to the node(s) specified by the **-n** *node* option. The **-n** *node* option specifies the ID or entry name of the node to which you are sending messages.

   Execute two commands because the first one generates network traffic and the second one measures the success of the traffic.

2. Examine output from the second **netstat** command. Monitor the results of this command for the remainder of the procedure.

   These lines give the node ID of the last node to detect a hardware failure and the date and time the failure was detected.

   Examples of iterations of the second **netstat** command follow.

```
**** Node 2603 ****  //budapest
Time 1987/03/06.13:54:16  Up since 1987/03/06.08:33:06

Net I/O:          total= 128696    rcvs = 126323    xmits =    2373

  888 page-in  requests issued.
  45 page-out requests issued.
  126 page-in  requests serviced.
  0 page-out requests serviced.
Detected concurrency violations -- read: 0     write: 0

 Xmit count       2373          Rcv eor            0
 NACKs              80          Rcv crc            3
 WACKs             345          Rcv timout        12
 Token inserted      1          Rcv buserr         0
 Xmit overrun        0          Rcv overrun        1
 Xmit Ack par        0          Rcv xmit-err     672
 Xmit Bus error      0          Rcv Modem err      0
 Xmit timout         0          Rcv Pkt error      0
 Xmit Modem err      0          Rcv hdr chksum     0
 Xmit Pkt error     21          Rcv Ack par        1

    Delay switched OUT.
```

Procedure 9-2 (Cont.)

```
Winchester I/O:  total=  29258   reads=  22428   writes=   6830

 Not ready          0         Contrlr busy       0
 Seek error         0         Equip check        0
 Drive time out     0         Overrun            0
 CRC error percentage: 0.00%

 Last ring hardware failure detected by node A9B3
   on 1987/03/06 at 13:54
   Node was not receiving clock signals.
System configured with 3.0 mb of memory.
A total of 0 parity errors were detected.

.............waiting for 5 seconds...

**** Node 2603 ****  //budapest
Time 1987/03/06.13:54:21  Up since 1987/03/06.08:33:06

Net I/O:          total=     42   rcvs =     27   xmits =     15

Winchester I/O:   total=      2   reads=      2   writes=      0

.............waiting for 5 seconds...

**** Node 2603 ****  //budapest
Time 1987/03/06.13:54:26  Up since 1987/03/06.08:33:06

Net I/O:          total=     32   rcvs =     18   xmits =     14

Winchester I/O:   total=      0   reads=      0   writes=      0

.............waiting for 5 seconds...
```

If no node is identified, or if the time displayed is too long ago to be relevant, go to Step 8.

3. Switch out the loop that contains the node listed in the **/com/netstat -l** command.

4. See if the network is back up by watching the Xmit and Rcv error counts displayed by the **netstat** command. If these counts go to 0, the network is functional.

   If your network is functional, you have isolated the loop that caused the problem. Leave this loop switched out of the network and go to the next procedure to determine the failing section of the loop.

5. If the error counts remain high, study your network diagram. Determine if the node reported in the **netstat** output is at the start or end of a loop.

6. If the node is at the either end of a loop, switch the upstream (for nodes at the start of a loop), or downstream (for nodes at the end of a loop), loop out of the network. See if the network is now functional by studying the **netstat** output.

Procedure 9–2 (Cont.)

7. If the network is now functional, try switching the loop that you switched out first, back into the network. If the network remains up, the fault is in the last loop you switched out. (A loop immediately upstream or downstream of the loop that contains the node listed in the **netstat** command.) Go to the next procedure to determine the cause of the fault in the failing loop.

   If the network fails again, the fault may be at the junction between the two loops. Carefully check the connections at the network switches for loose cables and go to the next procedure.

8. If the network is still down, switch all loops out of the network one loop at a time. After each loop is out, test to see if the network is back up. Once the network comes back up, switch the loops back into the network, one loop at a time, in the same order that you used to switch them out. Test after each loop to see if the network is back up. In this manner, you should isolate the problem to one or two loops. Go to the next procedure.

9. If all else fails, call your service representative.

END OF PROCEDURE 9–2

PROCEDURE 9-3: Locating the Point of Failure

This procedure describes locating the point of failure in a loop. To execute the commands described in this procedure, use a node in the loop that you assume to contain the failing node.

1. Use **netmain** on a node running a monitor in the loop that you isolated. Run the ERR_COUNTs probe so that it samples each node every few minutes.

2. On the same node, or on another node in this loop, execute the following two netstat commands, in separate shell processes (see Step 1 of Procedure 9-2).

   $ /com/netstat -l -r 5 -s 10 -n node <RETURN>

   and

   $ /com/netstat -l -r 5 <RETURN>

   The node listed in the output of the **netstat** commands should be either the node that caused the failure or the node next to the one that caused the failure. If a node is listed, move to Step 3. If no node is identified, or if the time displayed is too long ago to be relevant, move to Step 7.

3. Go to the node that detected the last hardware error as reported by the **netstat** command. Check all cables connected to this node to ensure that they are connected correctly. If any cables are loose, tighten and recheck the status of the loop. If the loop is now functional, you have corrected the problem. Switch the loop back in the network.

4. Use the **netsvc -n** command to remove the node reported to have received the last hardware error from the network. This command must be run on the node you're checking.

   $ netsvc -n <RETURN>

5. Examine the output of the **netstat** commands to see if the loop is now functional. If the network is functional, leave the node out of the network and go to Step 11. If the network is still failing, go to Step 6.

6. Go to the node that is immediately upstream of the node in Step 3. Follow the procedure described in Steps 3 through 5 on this node.

   If the network still is failing, perform the procedures described in Steps 3 through 5 on the node that is downstream from the one in Step 3.

7. If the network is still failing, use the Netmain Analyze Network Data menu. Select the following options and commands:

   - Select executing monitor (F2)
   - Error Values Plotted (use XMIT error)
   - Error Peaks output format
   - Top of error percentage scale (select 10 to 30%; then adjust if necessary)

8. Go to the node for which the highest error percentage is reported, and repeat the procedure's Steps 3 through 5.

9. If all else fails, shut down the nodes in the loop, one by one, by running the **netsvc -n** command on each one. After shutting down a node, check the status of the network. Work in one direction from the node in Step 3 around the network.

10. If you cannot isolate the failure, leave the loop switched out of the network and call your service representative.

11. Once you have isolated the failing node, begin to switch nodes back into the loop. Switch in one node at a time with the **netsvc -a** command and check the status of the network after switching in each node. If the network works, leave the node in. If it fails, switch it out of the loop. Once the loop is functional, switch the loop back into the network.

END OF PROCEDURE 9-3

Chapter 10

The Netmain Interactive Tool for Managing the Network

This chapter 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. The contexts in which to use **netmain** are described in Chapters 7, 8, and 9. Most of this chapter contains reference information on each of the **netmain** menus. This chapter also contains a tutorial, in case you have never used **netmain**.

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 end of this section. 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 on **netmain** before or after you practice with the tutorial.

You can configure **netmain_srvr** from the command line (see Chapter 6) 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 that can be 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 item in every menu.

Invoking netmain

Use the shell command **pst** to verify that a monitor is running before 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 <return>
```

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.

The netmain Top-Level Menu

When you invoke **netmain**, the program displays its top-level menu, shown in Figure 10–1. Use the top–level menu to

- Invoke the other menus
- Run **netmain** in an icon window

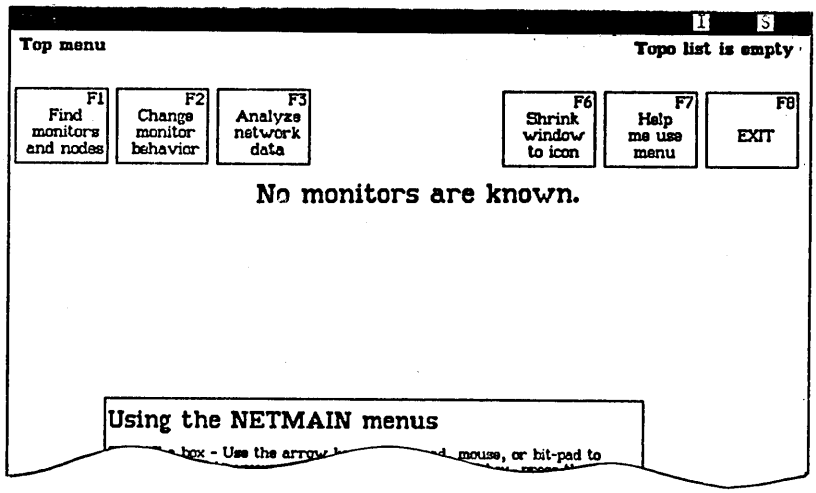


Figure 10–1. Top-Level Netmain Menu

Find Monitors and Nodes F1 Invokes a submenu that selects a single monitor from which to collect information or generate reports about monitors or nodes. Use the F1 menu to select monitors or to create and manipulate network topology lists.

Change Monitor Behavior F2 Invokes a submenu that redefines the **netmain_srvr** parameters. Use the F2 menu to change the behavior of an active monitor or to close an active log file for examination.

Analyze Network Data F3 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 the F3 menu to format data from an executing monitor or from a log file that has been closed.

Shrink Window to Icon F6 Runs the program in an icon window.

Help Me Use Menu F7 Provides online help.

Exit F8 Quits the program.

The netmain Find Monitors and Nodes Menu

Figure 10–2 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

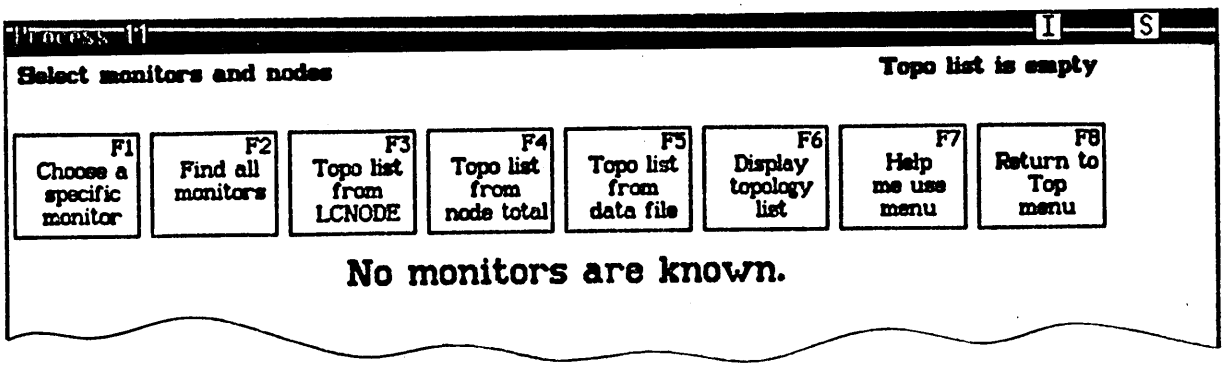


Figure 10–2. Find Monitors and Nodes Menu

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 10–1 explains each command box in the Find Monitors and Nodes menu.

Table 10–1. 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," and you want to examine information in the log file currently open for this monitor, to examine the monitor itself, or to reconfigure the monitor. Use the input box 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 • The monitor is in a loop that's switched out • A diskless node can't page information In case the monitor is busy, always try the command a second time if you receive this message. You cannot use other **netmain** features until you use this command or the F2 command to 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.

Table 10–1. Find Monitors and Nodes Commands (Cont.)

Task	Command Box Used	Explanation
Fill topology list with an estimated list 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 on-line list of all the nodes in the network. When the monitor has run for a week, the Total Node List should include all the 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 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 with a list of nodes from a file	F5 — Topo List from data file	Prompts for a filename and 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 or stored network topology kept in 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.

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 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 10–3 shows the Change Monitor Behavior menu, and Table 10–2 lists each command box in the menu.

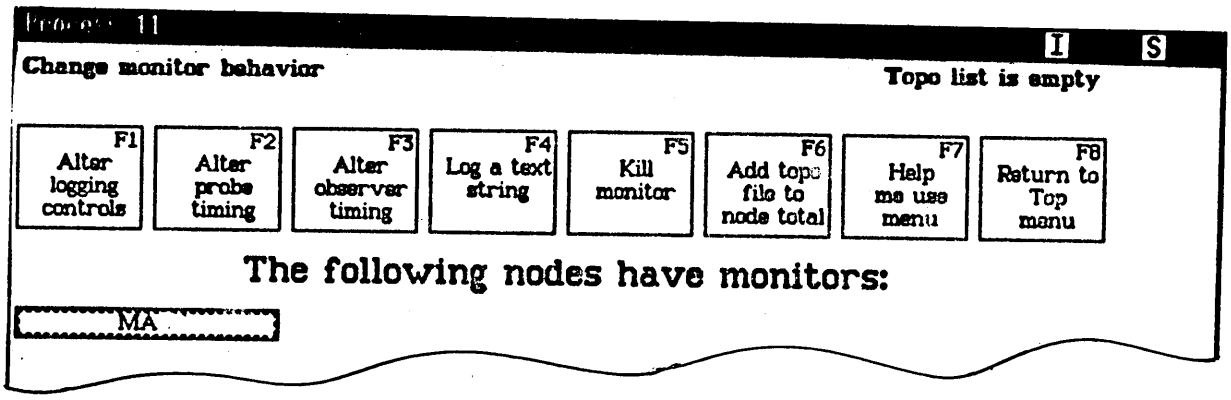


Figure 10–3. Change Monitor Behavior Menu

Table 10–2. Change Monitor Behavior Commands

Task	Command Box Used	Explanation
Set parameters for log file	F1 — Alter logging controls	Starts new log files, which allow you to change logging parameters.
Set parameters for probes, list probes	F2 — Alter probe timing	Reschedules probe active and waiting periods and probe skip distances.
Set parameters for observers, and list observer	F3 — Alter observer timing	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, and choose F2 — Find all monitors. Note that once you stop the monitor, you must restart it.
Update a monitor's total node list when the TOPOLOGY probe cannot	F6 — Add topo file to node total	Updates the chosen monitor's Total Node List 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 node 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.

The netmain Alter Logging Controls Submenu

The Alter Logging Controls submenu, shown in Figure 10–4, displays the parameters in effect for the log file currently in use. These parameters are the same as the following **netmain_srvr** options.

-l|-nl Write or do not write to a logfile

-l [pathname] The logfile name

-ll n Limit the size of the logfile

Use this menu to:

- Change the parameters in effect for a log file, or
- Cancel changes and return to the parameters most recently in effect

Table 10–3 lists each command box in the "Alter Logging Controls" submenu.

Table 10–3. 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 parameters	F2 — Revert to original logging	Causes changes made to return to the latest set of parameters actually being performed. When you use the Alter Logging Controls submenu, you are 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 Logging Controls Submenu

The number of option boxes that you see in the Alter Logging Controls submenu depends upon 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 10–4 shows the top sections of both menus.

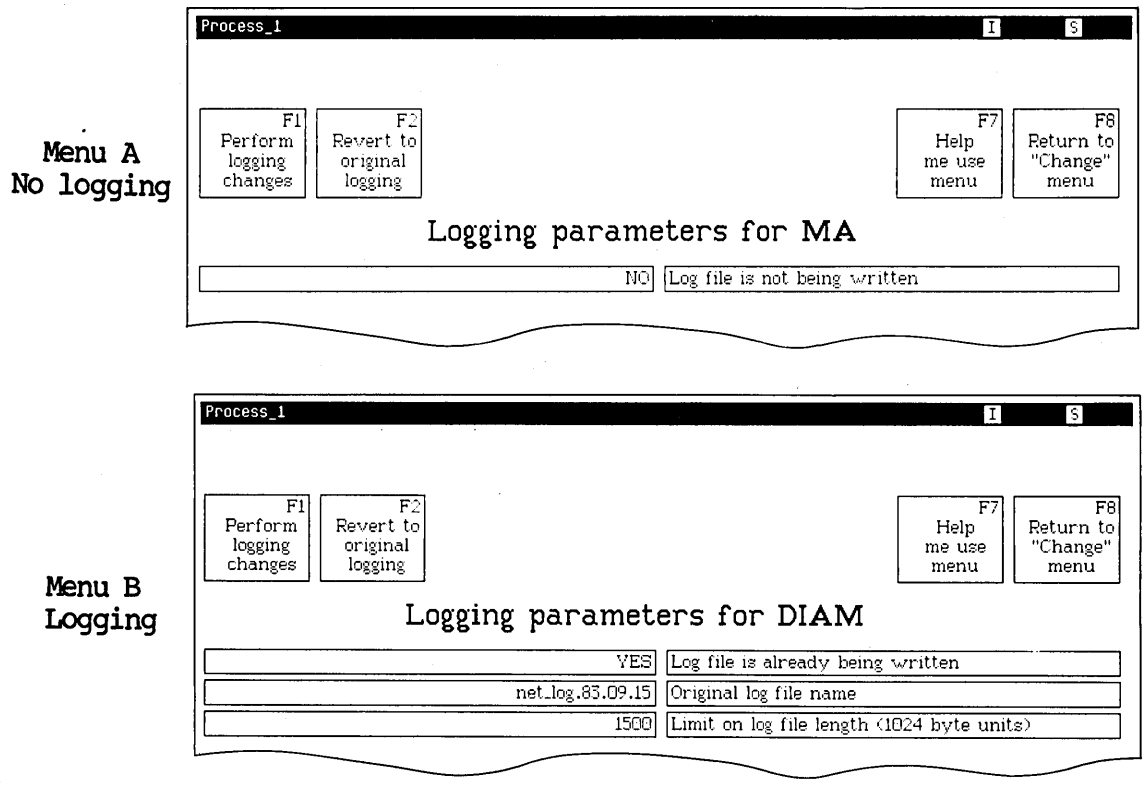


Figure 10–4. Alter Logging Controls Menu

In both versions, the left-hand boxes report the status of each parameter. The 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, New Log File will be written, to close a log file. The options at monitor start-up 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, 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/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 "Perform logging changes" or "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 10–5) 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 Chapter 6.

Table 10–4 lists each command box in the "Alter Probe Timing" submenu.

Table 10–4. The netmain Alter Probe Timing 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 parameters	F2 — Revert to original logging	Causes changes made to return to the 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 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 10–5, each probe name appears in the long box at the right; parameters for the probe are on the left. Figure 10–5 shows the top part of the Alter Probe Timing submenu.

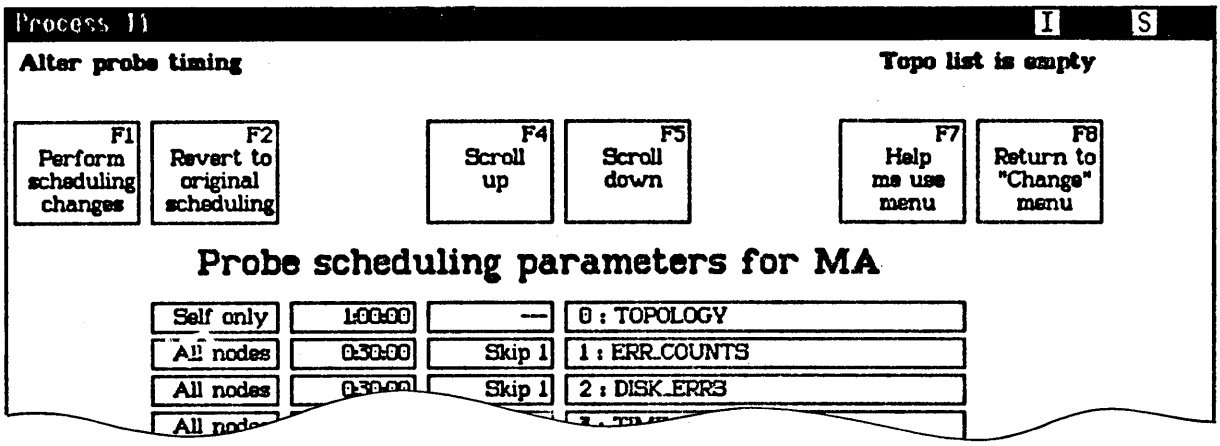


Figure 10–5. Alter Probe Timing Submenu

Table 10–5 lists the probe parameters appearing in the leftmost boxes of the Alter Probe Timing submenu.

Table 10–5. Probe Parameters

Parameter	Position	Description
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 Chapter 6.
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 10–5. Probe Parameters (Cont.)

Parameter	Position	Description
		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 "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.

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 **lcnode**s 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 – 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 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 10–6) 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 Chapter 6.

Table 10–6 lists each command box in the Alter Observer Timing submenu.

Table 10–6. The netmain Alter Observer Timing 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 parameters	F2 — Revert to original logging	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 the Alter Observer Timing menu (Figure 10–6) the rightmost box 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.

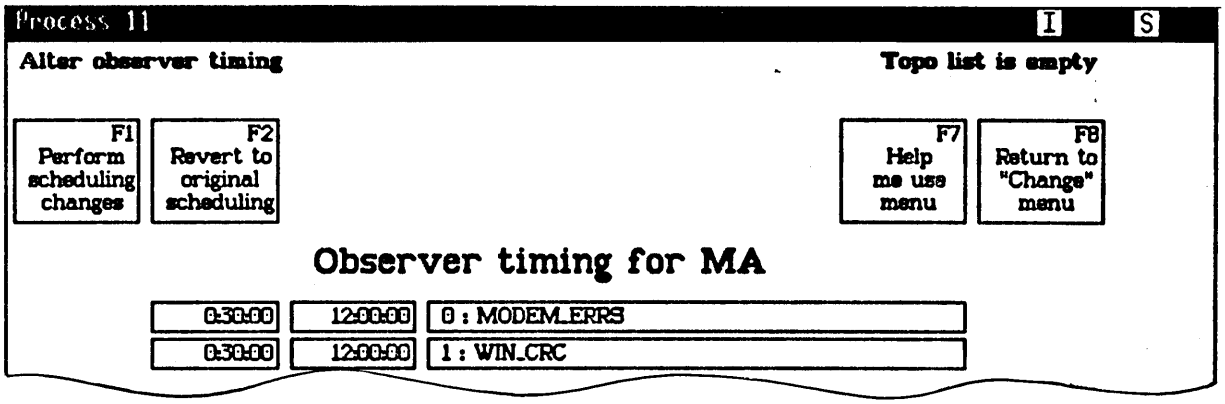


Figure 10–6. Alter Observer Timing Menu

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 10–7 shows the Analyze Network menu, and Table 10–7 lists each command box in the menu.

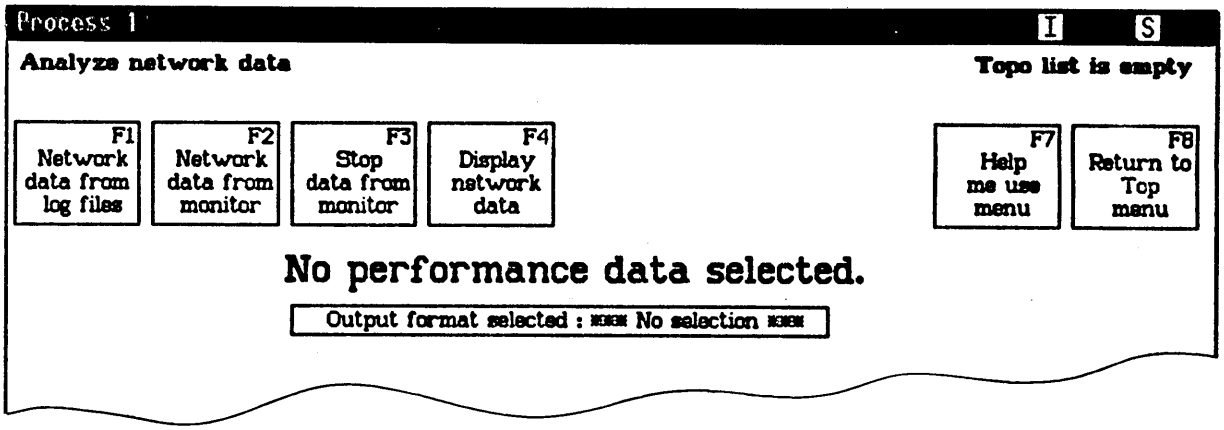


Figure 10–7. Analyze Network Data Menu

Table 10-7. The netmain Analyze Network Data Menu

Task	Command Box Used	Explanation
Select a log file	F1 — Network Data from log files	Brings up a menu 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 in 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 output format	F4 — Display Network Data	Formats the network data 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 which 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 sections 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 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 CALENDAR 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 submenu to reschedule the probes to their original sampling schedule. You can start another log file.

Choosing Output Formats for Data

This section 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 the Output Format Selected command 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 10–8 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 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. Use the parameter selection menu to choose a category and a performance statistic within that category.

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 10–8. 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.
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.
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.**

Table 10–8. Output Format Descriptions (Cont.)

Format	Description
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.
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. 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.
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.
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 10–9 describes each parameter and shows the output formats for which it is required.

Table 10–9. Parameters for Output Formats

Parameter	Required By These Output Formats	Parameter Function
Select a log file	F1 — Network Data from log files	Brings up a menu of log files you can analyze. If multiple log files are specified, **netmain** merges the contents of the files before performing the analysis.
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 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.

Table 10–9. Parameters for Output Formats (Cont.)

Parameter	Required By These Output Formats	Parameter Function
Top of error scale	Bar chart error Density peaks, 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. 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, Peaks Across net, Density Plots (or peaks)	Specifies the data value you see in plots and scattergrams. Displays a submenu from which you choose one of the categories shown in Table 10–8. See Chapter 7 for information about specific performance statistics.
Data dumped	Binary	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 Chapter 6 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.)
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 section). 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 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 <RETURN>
```

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 the <READ> or <EDIT> key. 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 *DOMAIN System Command Reference* manual for more information about the **prf** and **cpscr** commands.

Getting Started with netmain

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 previous sections for a full description of the Netmain Interactive Tool menus and to Chapter 8 for a discussion of how to use the tool for network management.

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 in Chapter 6. Let the monitor run, with default values, for a week. Then you may begin the procedures described in this section.

Procedure 10-1. Getting Started with netmain

1. Invoke **netmain** by typing

   $ netmain -whelp <return>

   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.

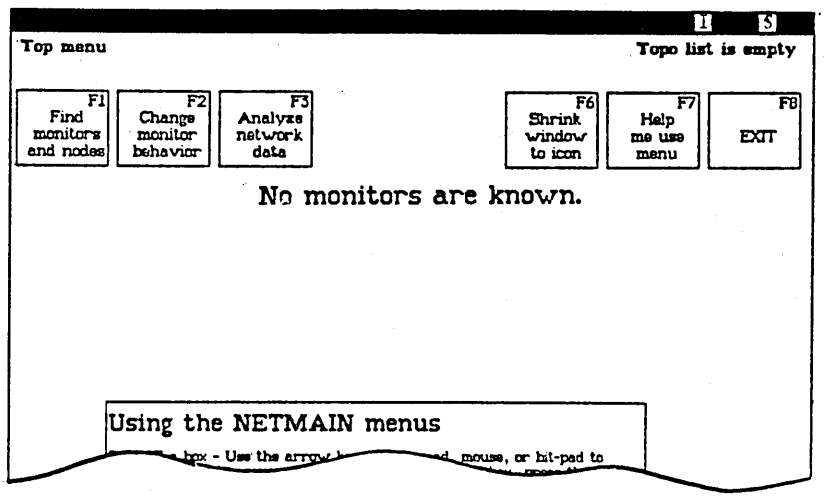


Figure 10-8. Top-Level Menu

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.

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.

4. Move the cursor back to the **netmain** window.

   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.

Procedure 10-1 (Cont.)

5. Point to the F6 Shrink window to make the command box 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.

6. Select the F6 Shrink Window to Icon command. To select a box in a **netmain** menu, you have the following choices:

   - 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. Select the F6 Shrink window command box using any of these methods.

7. 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. Point to the **netmain** icon by moving the DM cursor over the icon.

8. Request help about the icon by pointing to it and typing **h**, **?**, pressing the middle or right buttons on the mouse, or pressing any button except the top on the bit-pad puck. A system help file is displayed.

9. Enter **netmain** by pointing to the icon and pressing the space bar. Then select F2 Change monitor behavior, and note the error message that appears. You have not yet selected a monitor with which to work. You'll do this in the next procedure.

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**.

END OF PROCEDURE 10-1

Execute this procedure at a node where a **netmain_srvr** monitor is running.

PROCEDURE 10-2: Using the Monitor Location and Control Menus

1. Invoke **netmain** again. From the Top Level menu, select command box F1 to display the Find Monitors and Nodes menu. The top section of this menu is shown below.

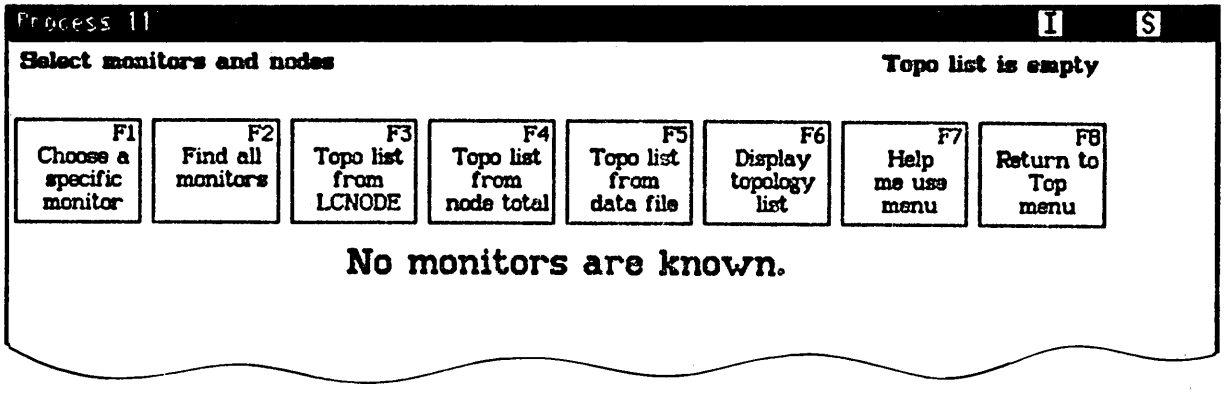


Figure 10-9. Find Monitors and Nodes Menu

2. Select F1 in this menu to choose a specific monitor. An **input box** containing a triangular cursor appears below the command boxes. Help for the input box also appears. Whenever an input box and a triangular cursor appears, you must enter something in the input box. Often, you can type <RETURN> to select the default behavior for the input box.

3. Type <RETURN> to select the monitor running on this node. The status message "Using this node" appears above the command boxes while **netmain** looks for the monitor.

4. The next **netmain** status message reports "The following nodes have monitors" and displays the name of this node in a box. If **netmain** reports "Node appears not to have a monitor," the monitor may have been busy and did not answer. Repeat Steps 2 and 3. Restart the monitor, using the information in Chapter 6 if the monitor still does not answer.

5. Select the F3 Topo list from **lcnode** command box. **Netmain** displays a status message as it executes the **lcnode** shell command to find the network topology. **Netmain** stores the list of all the nodes in the network in a topology list. After **netmain** fills the topology list, the topology list status message changes. It describes the new contents of the topology list.

6. Select the F6 Display topology list command box to display the contents of the topology list. The display that appears shows all the nodes that responded to the **lcnode** command.

7. Select F8 to return to the Top Level menu. Then select the F2 Change Monitor Behavior command box to display the Change Monitor Behavior menu. Figure 10-10 shows the top section of the menu.

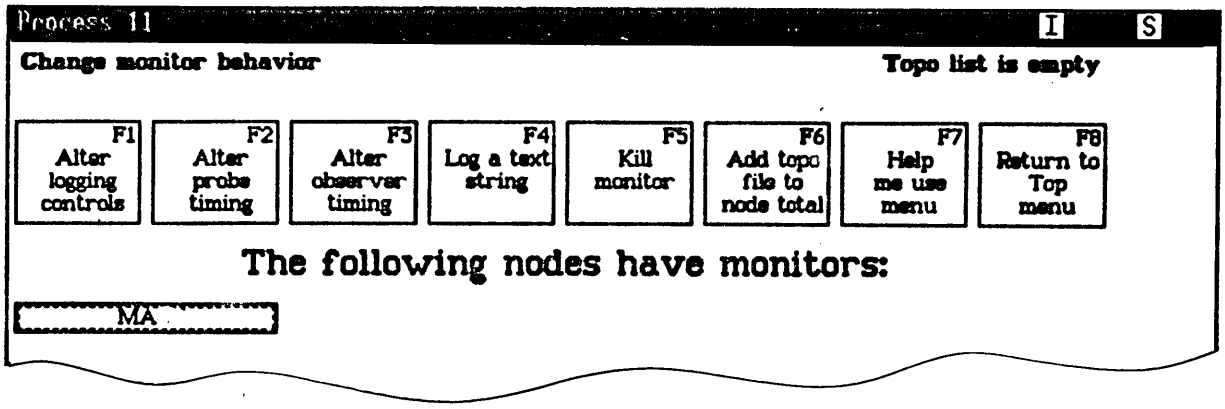


Figure 10–10. Change Monitor Behavior

Note that the monitor list and topology list status messages remain on the screen.

8. Select F1, the Alter Logging Controls submenu. The Change Monitor Parameter submenu appears. It describes the log file parameters for this monitor and allows you to change them.

   Notice the two command boxes, labeled F1 Perform logging changes, and F2 Revert to original logging. No change you make is final until you select F1 Perform logging changes. Select F2 Revert to original logging, to cancel a change that you entered in the input box but haven't put in effect. The function boxes, Logging parameters for node_name, show the status of each parameter on the left and the name of the parameter on the right.

9. Select the Log file is already being written parameter box to close the log file. When **netmain** reports "Log file will not be written," you've closed the log file.

10. Complete the change by selecting F1 Perform logging changes, then select F8 to Return to Change menu.

11. When you return to the Change Monitor Behavior menu, select F2 Alter Probe Timing to display a submenu of probes and probe scheduling parameters. Locate the input box for the Topology probe, and note the input box to the left of the probe box. It displays a default schedule for the probe (1:00:00). This means that the probe operates every hour.

13. Select the probe scheduling input box. In the input box that appears, enter "2:00:00" to change the scheduling for this probe to every two hours.

14. Cancel the change you've made by choosing F2 Revert to original scheduling. The monitor behavior submenus always allow you to cancel a change you've made.

15. Select F8 to return to the Change Monitor Behavior menu. From this menu, select F8, Return to Top menu. Then select F1, Find Monitors and Nodes menu.

16. Select the F5 Topo list from data file command box. The input help box describes how **netmain** resolves relative pathnames, if you use one. Enter the logfile pathname in the input box to write the contents of the log file, which you closed in Step 10, to the Topology List. You will use this file in the next procedure to generate **netmain** output.

END OF PROCEDURE 10-2

Procedure 10-3. Using the "Analyze Network Data" Menu

1. Invoke **netmain** and select the F3 command box to display the Analyze Network Data menu. Figure 10-11 shows the top section of the menu. Choose F1 Network data from log files.

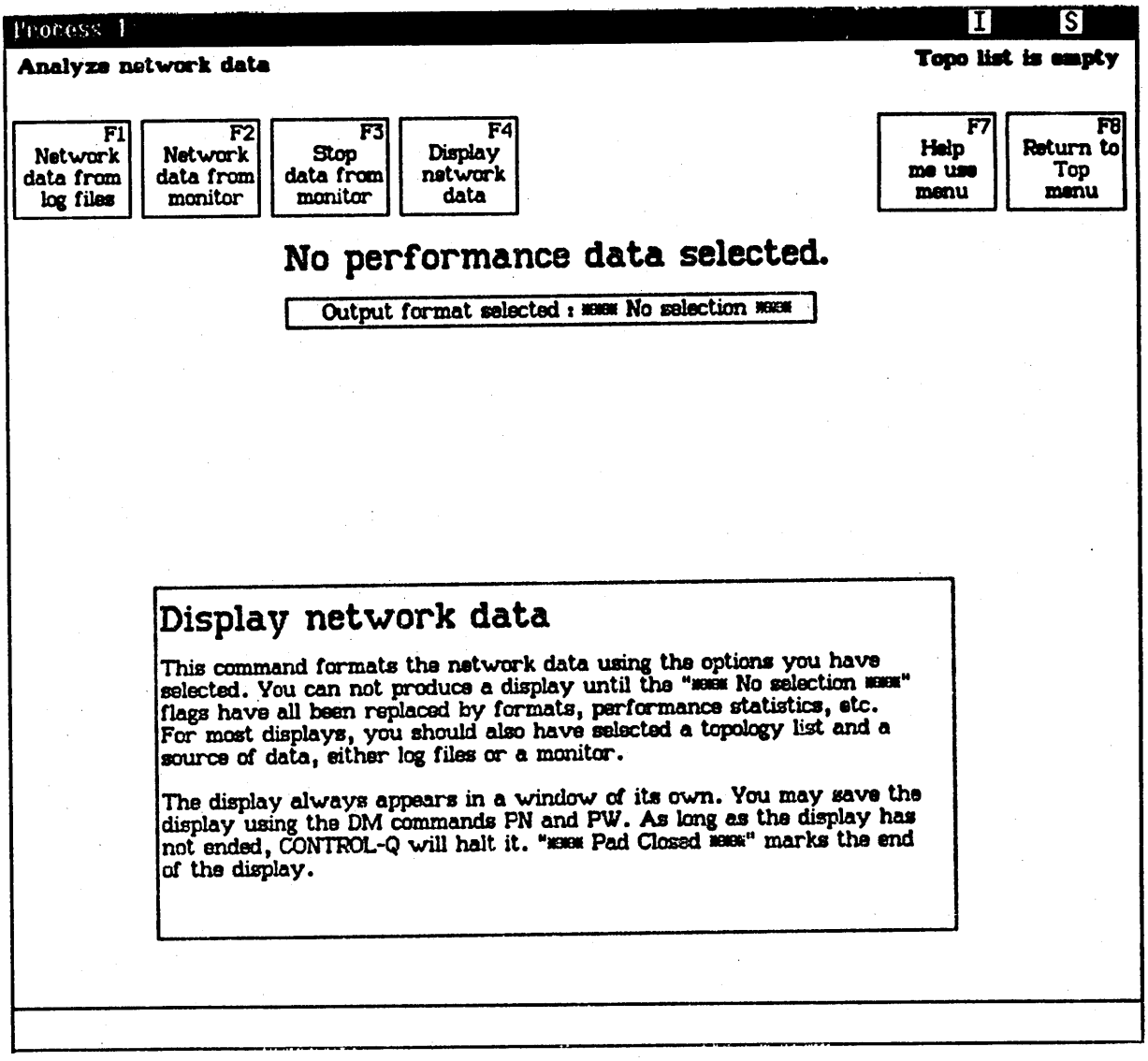


Figure 10-11. Analyze Network Data Menu

2. In the submenu that appears, select one of the long rectangular boxes. (There is a small box to the right of the long box; the small box contains dashes indicating no selection.) When an input box appears enter the pathname of the log file you closed in the last procedure. It should look like this.

```
`node_data/net_log/net_log.yy.mm.dd
```

   This is the file you put in the topology list in the last procedure. A status message, below the command boxes, names the file you select. Return to the Analyze Network Data menu.

3. Select the "Output format selected" box (the long rectangular box at the bottom of the menu). In the output format menu that appears, select the "Incidence density" option.

4. Parameter selection options appear beneath the output format box. Most parameters have default values. Choose "Data plotted," directly under the "Output format selected" box. It does not have a default selection.

5. From the "Select performance data" menu items that appear, choose the RING RECEIVE category. Notice the individual performance statistics that appear beneath the performance data categories.

6. Choose the "Rcv CRC" performance statistic.

7. Reset the value for the "Top of error scale" parameter, at the bottom of the parameter list. Set the value to 1%. You're now ready to display data from the sample log file.

8. Select F4 Display network data, to produce an incidence density plot from the log file. **netmain** produces a gray scale plot. It may take several minutes for the plot to appear.

9. Scroll horizontally through the plot, as **netmain** continues to produce it, and notice the areas of high and low density. Areas of high density indicate high levels for the "Rcv CRC" performance statistic. Chapter 7 describes how to draw conclusions about your network's behavior from gray scale plots like this.

10. Type CTRL/N to close the pad for the plot. If **netmain** was still producing output into the plot, it will display a message stating that the output was aborted. You can used the "PN" Display Manager command to name the pad, and save it, which **netmain** would also report in a message. Chapter 9 describes how to save **netmain** displays.

11. You may exit from **netmain** or practice with other kinds of displays and performance statistics.

END OF PROCEDURE 10-3

Index

The letter *f* means "and the following page"; the letters *ff* mean "and the following pages". Symbols are listed at the beginning of the index.

P

R

S

T

U