Compaq TCP/IP Services for OpenVMS

Management

Order Number: AA–LU50L–TE

January 2001

This manual describes how to configure and manage the TCP/IP Services product.

Revision/Update Information:	This manual supersedes *Compaq TCP/IP Services for OpenVMS Management*, Version 5.0.
Software Version:	Compaq TCP/IP Services for OpenVMS Version 5.1
Operating System:	OpenVMS Alpha Versions 7.1, 7.2-1 OpenVMS VAX Versions 7.1, 7.2

Compaq Computer Corporation
Houston, Texas

ZK6526

This document is available on CD-ROM.

This document was prepared using DECdocument, Version 3.3-1b.

Contents

4 Configuring Routing

Part 2 BIND

5 Configuring and Managing BIND

6 Using DNS to Balance Work Load

Part 3 Configuring Services

7 Configuring the DHCP Server

8 Configuring the DHCP Client

9 Configuring BOOTP

13 Configuring SNMP

Part 4 Configuring Network Applications

14 Configuring and Managing TELNET

15 Configuring and Managing FTP

16 Remote (R) Commands

17 Configuring and Managing SMTP

18 Configuring and Managing the POP Server

19 Configuring XDMCP-Compatible X Displays

Part 5 Network File Services

20 NFS Server

23 Setting Up and Managing TELNETSYM

24 Setting Up PC-NFS

Part 7 Appendixes

A Gateway Routing Daemon (GATED) Configuration Reference

B EBCDIC/DMCS Translation Tables

C How NFS Converts File Names

D Acronyms

Index

Examples

Tables

Preface

The Compaq TCP/IP Services for OpenVMS product is the Compaq implementation of the TCP/IP networking protocol suite and internet services for OpenVMS Alpha and OpenVMS VAX systems.

A layered software product, TCP/IP Services provides a comprehensive suite of functions and applications that support industry-standard protocols for heterogeneous network communications and resource sharing.

This manual provides system and network managers with information needed for the day-to-day management of the TCP/IP Services software product. This manual is best used in conjunction with the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.

See the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual for information about installing, configuring, and starting this product.

Intended Audience

This manual is for experienced OpenVMS and UNIX system managers and assumes a working knowledge of TCP/IP networking, TCP/IP terminology, and some familiarity with the TCP/IP Services product.

Document Structure

This manual contains seven parts, as follows:

Part 1 Describes how to configure network interfaces, how to set up serial lines, and how to configure and manage routing.

Part 2 Describes how to set up and manage the BIND server, resolver, and load broker components.

Part 3 Describes how to set up the following network services:

DHCP server
DHCP client
BOOTP and TFTP
Portmapper
Network Time Protocol (NTP)
SNMP

Part 4 Describes how to configure network applications that let users send and receive electronic mail from the internet, establish login sessions with a remote host, and transfer files. These network applications are:

TELNET
FTP
Remote (R) commands
SMTP and POP
XDM-compatible X displays

Part 5 Describes how to configure, use, and manage the components that enable transparent network file sharing, including the NFS server and NFS client.

Part 6 Describes how to configure and manage network printing services, including LPD/LPR, TELNETSYM, and PC-NFS.

Part 7 Provides appendixes that:

- Explain how to configure GATED.
- Provide EBCDIC/DMCS translation tables.
- Describe how NFS converts UNIX file names to OpenVMS files names.
- List the acronyms related to TCP/IP networking.

Related Documents

Table 1 lists the documents available with this version of TCP/IP Services.

Table 1 TCP/IP Services Documentation

Manual	Contents
DIGITAL TCP/IP Services for OpenVMS Concepts and Planning	This manual provides conceptual information about networking and the TCP/IP protocol including a description of the Compaq implementation of the Berkeley Internet Name Domain (BIND) service and the Network File System (NFS). It outlines general planning issues to consider before configuring your system to use the TCP/IP Services software. This manual also describes the manuals in the documentation set, provides a glossary of terms and acronyms for the TCP/IP Services software product, and documents how to contact the InterNIC Registration Service to register domains and access Request for Comments (RFCs).
Compaq TCP/IP Services for OpenVMS Release Notes	This text file describes new features and changes to the software including installation, upgrade, configuration, and compatibility information. These notes also describe new and existing software problems and restrictions, and software and documentation corrections. Print this text file at the beginning of the installation procedure and read it before you install TCP/IP Services.
Compaq TCP/IP Services for OpenVMS Installation and Configuration	This manual explains how to install and configure the TCP/IP Services product.

(continued on next page)

Table 1 (Cont.) TCP/IP Services Documentation

Manual	Contents
DIGITAL TCP/IP Services for OpenVMS User's Guide	This manual describes how to use the applications available with TCP/IP Services such as remote file operations, email, TELNET, TN3270, and network printing. This manual explains how to use these services to communicate with systems on private internets or on the worldwide Internet.
Compaq TCP/IP Services for OpenVMS Management	This manual describes how to configure and manage the TCP/IP Services product. Use this manual with the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.
Compaq TCP/IP Services for OpenVMS Management Command Reference	This manual describes the TCP/IP Services management commands. Use this manual with the *Compaq TCP/IP Services for OpenVMS Management* manual.
Compaq TCP/IP Services for OpenVMS Management Command Quick Reference Card	This reference card lists the TCP/IP management commands by component and describes the purpose of each command.
Compaq TCP/IP Services for OpenVMS UNIX Command Reference Card	This reference card contains information about commonly performed network management tasks and their corresponding TCP/IP management and Compaq *Tru64* UNIX command formats.
DIGITAL TCP/IP Services for OpenVMS ONC RPC Programming	This manual presents an overview of high-level programming using open network computing remote procedure calls (ONC RPC). This manual also describes the RPC programming interface and how to use the RPCGEN protocol compiler to create applications.
Compaq TCP/IP Services for OpenVMS Sockets API and System Services Programming	This manual describes how to use the Sockets API and OpenVMS system services to develop network applications.
Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference	This manual describes the Simple Network Management Protocol (SNMP) and the SNMP application programming interface (eSNMP). It describes the subagents provided with TCP/IP Services, utilities provided for managing subagents, and how to build your own subagents.
Compaq TCP/IP Services for OpenVMS Tuning and Troubleshooting	This manual provides information about how to isolate the causes of network problems and how to tune the TCP/IP Services software for the best performance.
Compaq TCP/IP Services for OpenVMS Guide to IPv6	This manual describes the IPv6 environment, the roles of systems in this environment, the types and function of the different IPv6 addresses, and how to configure TCP/IP Services to access the 6bone network.

For additional information about Compaq *OpenVMS* products and services, access the Compaq website at the following location:

`http://www.openvms.compaq.com/`

For a comprehensive overview of the TCP/IP protocol suite. you might find the book *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, by Douglas Comer, useful.

Reader's Comments

Compaq welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@compaq.com**
Mail	Compaq Computer Corporation OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How to Order Additional Documentation

Visit the following World Wide Web address for information about how to order additional documentation:

`http://www.openvms.compaq.com/`

If you need help deciding which documentation best meets your needs, call 800-282-6672.

Conventions

The name TCP/IP Services means both:

- Compaq TCP/IP Services for OpenVMS Alpha
- Compaq TCP/IP Services for OpenVMS VAX

The name UNIX refers to the Compaq *Tru64* UNIX operating system.

The following conventions are used in this manual. In addition, please note that all IP addresses are fictitious.

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.) In the HTML version of this document, this convention appears as brackets, rather than a box.
. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.

( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
|	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold text	This typeface represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace text`	Monospace type indicates code examples and interactive screen displays. This typeface indicates UNIX system output or user input, commands, options, files, directories, utilities, hosts, and users. In the C programming language, this typeface identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

Part 1

Connecting to the Network

Part 1 provides the information on how to get started after installing and configuring the TCP/IP Services software.

Part 1 includes the following chapters:

- Chapter 1, Managing TCP/IP Services, describes the management control interfaces that allow you to configure and manage TCP/IP Services.
- Chapter 2, Configuring Interfaces, describes how to set up network interfaces.
- Chapter 3, Configuring Serial Lines, explains how to set up serial lines.
- Chapter 4, Configuring Routing, discusses how to configure and manage network routing.

1
Managing TCP/IP Services

This chapter reviews information you need to get started with the TCP/IP Services software. Topics include:

- Reviewing pertinent databases, logical names, and configuration guidelines (Section 1.1).
- Enabling support for DECnet over TCP/IP, and PATHWORKS (Advanced Server) (Section 1.2).
- Creating user accounts and proxy identities (Section 1.3).
- Configuring TCP/IP Services on an OpenVMS cluster (Section 1.4).
- Starting services with the auxiliary server (Section 1.5).

1.1 Getting Started

This manual assumes you installed and configured TCP/IP Services software with the configuration procedure SYS$MANAGER:TCPIP$CONFIG.COM, as described in the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual. This menu-driven procedure configures the software components you select or all of the TCP/IP Services software components. The "out-of-the-box" defaults are designed to get your system up and running as an internet host with minimal effort.

TCPIP$CONFIG creates the database files listed in Table 1–1.

Table 1–1 Configuration Databases

Database	File Name
BOOTP database	SYS$COMMON:[SYSEXE]TCPIP$BOOTP.DAT
Configuration database	SYS$COMMON:[SYSEXE]TCPIP$CONFIGURATION.DAT
Export database	SYS$COMMON:[SYSEXE]TCPIP$EXPORT.DAT
Hosts database	SYS$COMMON:[SYSEXE]TCPIP$HOST.DAT
Networks database	SYS$COMMON:[SYSEXE]TCPIP$NETWORK.DAT
Proxy database	SYS$COMMON:[SYSEXE]TCPIP$PROXY.DAT
Routes database	SYS$COMMON:[SYSEXE]TCPIP$ROUTE.DAT
Services database	SYS$COMMON:[SYSEXE]TCPIP$SERVICE.DAT

1.1.1 Logical Names

Logical names allow you to customize or modify component behavior. Logical names also point to directories, database files, and log files.

TCPIP$CONFIG defines the following logical names for the databases listed in Table 1–1:

- TCPIP$BOOTP
- TCPIP$CONFIG
- TCPIP$EXPORT
- TCPIP$HOST
- TCPIP$NETWORK
- TCPIP$PROXY
- TCPIP$ROUTE
- TCPIP$SERVICE

See individual component chapters in this manual for information on how specific components use logical names.

1.1.2 Modifying Your Configuration

After the initial configuration, you may want to reconfigure existing components or configure new ones, disable and re-enable components, add hosts, reconfigure routing, and so forth.

When making any configuration modifications, Compaq strongly recommends that you run the configuration procedure TCPIP$CONFIG again.

Note

You cannot use TCPIP$CONFIG to set up SLIP or PPP lines. See Chapter 3 for more information.

In some instances, TCPIP$CONFIG only partially configures a component (for example, when configuring a BIND name server). You may need to run additional setup programs or enter management commands to complete the configuration and fine-tune your environment.

Component-specific chapters in this manual describe additional configuration tasks and explain how to configure and manage specific components. These tasks may include:

- Manually adding information, such as database records, that the configuration procedure cannot handle
- Temporarily enabling or disabling a service
- Configuring customized applications
- Tuning performance
- Troubleshooting

1.1.3 Saving Changes

The configuration procedure TCPIP$CONFIG saves configuration and initialization information in the file TCPIP$CONFIGURATION.DAT. You can modify the configuration dynamically or permanently, as follows:

- SET commands modify the software dynamically, as it is running. Changes made in this manner are not saved permanently and are overwritten if they differ from settings in the permanent configuration database.
- SET CONFIGURATION commands modify the permanent database but do not take effect until the next time the product starts up.

To make changes take effect immediately and modify permanent settings, enter both the interactive SET and permanent SET CONFIGURATION commands.

The following commands permanently modify the configuration database:

- SET CONFIGURATION BIND
- SET CONFIGURATION COMMUNICATION
- SET CONFIGURATION ENABLE SERVICE
- SET CONFIGURATION DISABLE SERVICE
- SET CONFIGURATION INTERFACE
- SET CONFIGURATION NAME_SERVICE
- SET CONFIGURATION PROTOCOL
- SET CONFIGURATION SMTP
- SET CONFIGURATION SNMP
- SET CONFIGURATION START ROUTING
- SET CONFIGURATION START NOROUTING

Note

Throughout this manual, all commands are assumed to be TCP/IP management commands. Any DCL commands that are mentioned are identified as such.

For a full description of the TCP/IP management commands and a discussion of how to use them, see the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.

1.1.4 Starting and Stopping the Software

To start TCP/IP Services manually, enter the following command:

```
$ @SYS$STARTUP:TCPIP$STARTUP
```

The startup procedure enables the configured services and initializes the configured network interfaces.

To stop (shut down) the product manually, enter the following command:

```
$ @SYS$STARTUP:TCPIP$SHUTDOWN
```

The shutdown procedure does the following:

1. Stops network communication

2. Disables active services
3. Deletes the network interface definitions
4. Deassigns defined logical names
5. Deletes installed images

To start TCP/IP Services automatically, add the following command to the system startup file:

```
$ @SYS$STARTUP:TCPIP$STARTUP.COM
```

To maintain site-specific startup and shutdown commands and settings, create the following files:

- SYS$STARTUP:TCPIP$SYSTARTUP.COM
- SYS$STARTUP:TCPIP$SYSHUTDOWN.COM

The site-specific startup procedure is invoked after all the TCP/IP services have been started. These files are not overwritten when you install a new version of TCP/IP Services.

Compaq recommends that you use the TCPIP$CONFIG configuration procedure to stop and start services. However, startup and shutdown files are provided for individual services, allowing you to stop and start individual components without impacting the operation of the remaining TCP/IP Services software.

This feature allows you to modify a service configuration without restarting the TCP/IP Services product. For example, you can shut down the LPD service, change its configuration parameters, and then restart it, without interrupting the other TCP/IP services that are running on the system.

Each service is provided with its own startup and shutdown command procedures, as follows:

- SYS$STARTUP:TCPIP$*service*_STARTUP.COM, a supplied command procedure that ensures the environment is configured appropriately and starts up the component specified by *service*.
- SYS$STARTUP:TCPIP$*service*_SHUTDOWN.COM, a supplied command procedure that shuts down a specific service component without affecting the other services that are running.

To preserve site-specific parameter settings and commands for a specific service, create the following files, specifying the service or component name for *service*. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$*service*_SYSTARTUP.COM can be used to store site-specific startup commands.

  This procedure is invoked by the appropriate service-specific startup procedure prior to running the service. Use the *_SYSTARTUP procedure to modify the behavior of the service each time the service or TCP/IP Services is restarted. For example, to enable debugging mode for DHCP, define the logical TCPIP$DHCP_DEBUG in the SYS$STARTUP:TCPIP$DHCP_SYSTARTUP.COM file. When DHCP next starts, it will run in debug mode.
- SYS$STARTUP:TCPIP$*service*_SYSHUTDOWN.COM can be used to store site-specific shutdown commands.

Service-specific startup and shutdown procedures, as well as configuration parameters, are described in the later chapters of this manual.

1.2 Enabling PATHWORKS/Advanced Server and DECnet-over-TCP/IP Support

TCP/IP Services software includes the PATHWORKS Internet Protocol (PWIP) driver and the PWIP ancillary control process (PWIP_ACP).

The PWIP driver allows OpenVMS systems that are running both the Compaq PATHWORKS/Advanced Server and the TCP/IP Services software to communicate with personal computers running PATHWORKS client software. It also enables the DECnet-over-TCP/IP feature, which is included with the DECnet-Plus for OpenVMS Version 6.0 and later software. For more information about DECnet over TCP/IP, see the DECnet-Plus for OpenVMS documentation.

1.2.1 Starting and Stopping the PWIP Driver

The PWIP driver can be shut down and started independently. The following files are provided:

- SYS$STARTUP:TCPIP$PWIP_DRIVER_STARTUP.COM allows you to start up the PWIP driver.
- SYS$STARTUP:TCPIP$PWIP_DRIVER_SHUTDOWN.COM allows you to shut down the PWIP driver.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services.

- SYS$STARTUP:TCPIP$PWIP_DRIVER_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the PWIP driver is started.
- SYS$STARTUP:TCPIP$PWIP_DRIVER_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the PWIP driver is shut down.

To start the PWIP driver, run TCPIP$CONFIG or enter the following command:

```
$ @SYS$STARTUP:TCPIP$PWIP_DRIVER_STARTUP.COM
```

To shut down the connection to the PWIP driver, enter the following command:

```
$ @SYS$STARTUP:TCPIP$PWIP_DRIVER_SHUTDOWN.COM
```

1.3 Setting Up User Accounts and Proxy Identities

You will need to set up accounts for local users, coordinate the establishment of corresponding accounts on remote systems, and create accounts for remote users who will be accessing server components on the local host.

When creating accounts for remote users, you can create one account for all remote users, an account for groups of remote users, or accounts for individual users. The strategy you use depends on your organization, system resources, and security needs.

Certain product components (for example, LPD, RSH, RLOGIN, and NFS) act as servers for remote clients. You control access to your system and to these services by giving remote users proxy identities. A proxy identity maps a user account on one host to an account on another host. The information you provide with each entry, along with the privileges you set for the account, lets you specifically grant or deny access to your system.

The configuration procedure TCPIP$CONFIG creates a proxy database file called TCPIP$PROXY. You add proxies to this database with the ADD PROXY command. The TCP/IP Services product allows two types of proxies:

- Communication proxy

  A communication proxy provides an identity for remote users of RSH, RLOGIN, RMT/RCD, and LPD. For each host, be sure to define the host name and any aliases. Proxy entries are case sensitive. Be sure to use the appropriate case when adding entries for remote users. Enter the ADD PROXY command as follows:

  ```
  TCPIP> ADD PROXY user /HOST=host /REMOTE_USER=user
  ```

  You can use wildcards when adding proxy entries for users on remote systems. For example, the following command provides the identity STAFF to any user on the remote host STAR:

  ```
  TCPIP> ADD PROXY STAFF  /HOST=STAR /REMOTE_USER=*
  ```

- NFS proxy

  NFS proxies provide identities for users of NFS client, NFS server, and PC-NFS. In addition to host and user information, NFS proxies provide UNIX identities with UID/GID pairs. NFS proxies can specify access to the NFS client or the NFS server, or both.

  For example, the following command provides the OpenVMS identity CHESTER for a local NFS client user with the UID/GID pair 23/34.

  ```
  TCPIP> ADD PROXY CHESTER /NFS=OUTGOING /UID=23 /GID=34 /HOST="orbit"
  ```

  This user can access remote files from the NFS server `orbit`.

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for a complete description of the ADD PROXY command. For a more complete discussion about UNIX style identities and how the NFS server and client use the proxy database, see Chapter 20.

1.4 Configuring a TCP/IP Cluster

If your host is part of an OpenVMS Cluster, you can use a cluster alias to represent the entire cluster or selected host members. In this case, the network sees the cluster as a single system with one name. Alternatively, you can configure clustering using a DNS alias, as described in Chapter 5.

Incoming requests are switched among the cluster hosts at the end of each cluster time interval (specified with the SET COMMUNICATION command).

Note

The cluster name is not switched from a host if there are any active TCP connections to the cluster interface on that host.

A remote host can use the cluster alias to address the cluster as a single host or the host name of the cluster member to address a cluster member individually.

All of the TCP/IP services support automatic failover and can be run on multiple nodes in an OpenVMS Cluster. For example, if more than one host in the cluster is running the NFS server, the cluster can appear to the NFS client as a single

host. For more information about configuring a specific service for cluster failover, refer to the chapter in this manual that discusses the particular service.

1.4.1 Setting Up an ARP-Based Cluster

Compaq strongly recommends using the configuration procedure TCPIP$CONFIG to configure a TCP/IP cluster. If you cannot run TCPIP$CONFIG, configure a TCP/IP cluster by completing the following steps:

1. Create the interfaces for all cluster members.
2. Interactively specify an ARP-based cluster alias (for example, ALLOFUS). Enter:

   ```
   TCPIP> SET INTERFACE QE0 /CLUSTER=ALLOFUS  /C_NETWORK=255.255.0.0 -
   _TCPIP> /C_BROADCAST=128.44.55.0
   ```

3. Make these settings permanent in the configuration database. Enter:

   ```
   TCPIP> SET CONFIGURATION INTERFACE QE0 /CLUSTER=ALLOFUS -
   _TCPIP> /C_NETWORK=255.255.0.0 /C_BROADCAST=128.44.55.0
   ```

   The interface changes take effect the next time the product starts up.
4. Add the cluster host name or the cluster IP address to the database of the host. Enter the same information you use with the SET INTERFACE command.
5. Change the interface parameters (specified with the SET INTERFACE command) only after deleting and re-creating an interface.
6. Set the cluster timer with the SET COMMUNICATION or SET CONFIGURATION COMMUNICATION command. For example, enter:

   ```
   TCPIP> SET COMMUNICATION /CLUSTER_TIMER=30
   ```

7. Optionally, direct traffic to a specific host by entering the following command:

   ```
   TCPIP> SET COMMUNICATION /CLUSTER_TIMER=0
   ```

   The host owns the cluster alias as long as there are active TCP connections using the alias until you either bring down the system or delete the network interface.

1.5 Auxiliary Server

The auxiliary server is the TCP/IP Services implementation of the UNIX internet daemon (`inetd`). In addition to standard `inetd` functions, the auxiliary server provides access control and event logging.

The auxiliary server listens continuously for incoming requests and acts as a master server for programs specified in its configuration file. The auxiliary server reduces the load on the system by invoking services only as they are needed.

1.5.1 How the Auxiliary Server Works

The auxiliary server listens for connections on the internet addresses of the services that its configuration file (TCPIP$SERVICES.DAT) specifies. When a connection is found, it invokes the server daemon for the service requested. Once a server is finished, the auxiliary server continues to listen on the socket.

When it receives a request, the auxiliary server dynamically creates a network process, obtaining user account information from one or all of the following sources:

- TCP/IP Services proxy account
- Services database
- Remote client
- Local OpenVMS user authorization file (UAF)

In addition, users requesting services at the client can include their user account information as part of the command line.

Once a process is created, the auxiliary server starts the requested service. All services except RLOGIN and TELNET must have access to their default device and directories and to the command procedures within them.

1.5.1.1 Rejecting Client Requests

The auxiliary server rejects client requests for the following reasons:

- The maximum number of simultaneous processes for the requested service has been reached.
- The request is from a host that is marked for rejection.
- There is a problem with the target account or directory.

1.5.1.2 Configuring the Auxiliary Server

The postinstallation configuration procedure, TCPIP$CONFIG, creates an entry in the services database (TCPIP$SERVICE.DAT) for each service you configure. If you need to modify your initial configuration, run TCPIP$CONFIG or use the SET SERVICE command.

The configuration file TCPIP$SERVICE.DAT includes information about the service name, the socket and protocol type associated with the service, the user name under which the service should run, and any special options for the service program.

Before you activate a service manually, configure the auxiliary server as follows:

1. Use the OpenVMS Authorize utility to create a restricted user account for the process. Use the following qualifiers when creating the account:
   - /NOINTERACTIVE
   - /NOBATCH
   - /NOREMOTE
   - /FLAGS=(RESTRICTED,NODISUSER,NOCAPTIVE)

   For more information about creating restricted accounts, see the OpenVMS system security documentation.
2. Provide user account information that can be used when the network process is created. Plan your requirements carefully before setting privileges, quotas, and priorities to user accounts.
3. Provide the network process name.

The auxiliary server builds the network process name from the character string in the services database. Enter this string with the SET SERVICE command:

```
TCPIP> SET SERVICE service /PROCESS_NAME=process
```

Note

For TELNET and RLOGIN, the process name is set by either the system or users.

4. Set the maximum number of server processes that can run simultaneously. This number should not exceed the maximum number of sockets allowed on the system. To set the maximum number of processes that can connect to a service at the same time, enter the following TCP/IP management command:

   ```
   TCPIP> SET SERVICE service-name /LIMIT=n
   ```

   In this command, *service-name* is the name of the service to which the connections will be limited, and *n* is the number of connections that will be accepted by the service at one time.

   To activate the change, disable the service using the DISABLE SERVICE command, and then enable it using the ENABLE SERVICE command.

5. Make sure that the protections in the systemwide SYLOGIN.COM file are set appropriately. If they are not, enter the following DCL command:

   ```
   $ SET PROTECTION=(W:RE) SYS$MANAGER:SYLOGIN.COM
   ```

6. To ensure that the services database has an entry for each service offered, enter the SHOW SERVICE command.

1.6 Enabling Services

The services you configured are enabled during the TCP/IP Services startup procedure. Afterwards, to initialize (enable) a service, enter the following command:

```
TCPIP> ENABLE SERVICE
```

The ENABLE SERVICE command immediately changes the running system. The SET CONFIGURATION ENABLE SERVICE command causes the services to be enabled the next time TCP/IP Services starts up.

To specify the type of socket, include the /PROTOCOL qualifier on the SET SERVICE command line. For example, to specify stream sockets, enter /PROTOCOL=TCP. To specify datagram sockets, enter /PROTOCOL=UDP.

The auxiliary server can set socket options for a requested service either before or during data communications. Some available options are:

- KEEPALIVE (for TCP communications)
- BROADCAST (for UDP communications)

To set the socket options, include the /SOCKET_OPTIONS qualifier on the SET SERVICE command.

1.6.1 Setting Up Event Logging

Event logging can help you manage the software. By default, user-defined services do not log events, but you can enable event logging for all or selected configured services. You can configure the product to log events to the operator's console, a log file, or both. To set up event logging, enter the following command:

```
SET SERVICE service-name /LOG_OPTIONS=ALL
```

For a list of all the logging options, see the SET SERVICE command description in the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.

Some product components provide additional event logging capabilities. See individual component chapters for more information.

2

Configuring Interfaces

OpenVMS systems running TCP/IP Services communicate with other internet hosts over a variety of physical media. Because TCP/IP is independent of the underlying physical network, IP addresses are implemented in the network software, not the network hardware. (See the *Compaq TCP/IP Services for OpenVMS Software Product Description* for a complete list of supported media.)

This chapter reviews key concepts and describes:

- How to configure network controllers (Section 2.2)
- How to configure network interfaces (Section 2.3)

2.1 Key Concepts

A **network controller** is the hardware connection between a computer system and a physical network. Controllers perform the packet channeling to and from the physical medium of your network, usually a cable.

The **network interface** is a logical network controller — a software component that communicates with your network software and the network controller.

For each interface, you can enable or disable the interface, set the subnet mask, and assign IP and broadcast addresses.

2.2 Configuring Network Controllers

TCP/IP Services automatically recognizes network controllers at startup. If you need to change the configuration (remove, modify, or add new network controllers to your system) after installing and configuring the product, follow the installation and configuration instructions that come with your hardware; then run TCPIP$CONFIG again. The TCP/IP Services software will recognize the new controller immediately, and will create new interfaces the next time the software starts up.

Note

Hardware installation and configuration instructions are specific for the various network controllers. Be sure to read the instructions provided with your new hardware before installing.

2.3 Configuring Network Interfaces

The TCP/IP Services product supports one local software interface for loopbacks and one or more physical network interfaces for each physical network controller.

The configuration procedure initially configures your network interfaces. Use the following commands if you need to redefine an interface or configure serial lines. See Chapter 3 for more information about configuring serial lines.

- SET INTERFACE
- SET NOINTERFACE
- SET CONFIGURATION INTERFACE
- SET CONFIGURATION NOINTERFACE

To display information, use the SHOW INTERFACE command; to disable an interface, use the SET NOINTERFACE command.

Note

If you are redefining an existing interface, enter the SET NOINTERFACE command before you enter the SET INTERFACE command.

2.3.1 Specifying the Interface

Interface names include the following information:

- One letter indicating the interface type

  Interface types indicate the type of controller. The following table shows the letters you can use to indicate each type of controller:

For this controller	Use this interface type
ATM	I, L
Ethernet	B, C, D, F, I, N, O, Q, R, S, W, X, Z
FDDI	A, C, F, Q, R, W
Token Ring	C, R
PPP/SLIP	S
Local (loopback)	L

- One letter indicating the interface class

For this controller	Use this interface class
ATM	F
Ethernet	E
FDDI	F
Token Ring	T
PPP	P
Serial	L

For this controller	Use this interface class
X25	X
Local (loopback)	O

- An integer indicating the controller number. Controller numbers are decimal numbers in the range of 0 through 25, corresponding to OpenVMS hardware controller letters A through Z. The default is 0.

Primary interfaces for Ethernet controllers have names in the range SE, SE0, SE1, SE2, . . . SE24, SE25.

Interfaces for PPP controllers have names in the range PP, PP0, PP1, . . . PP998, PP999.

Interfaces for local (loopback) controllers have names in the range LO, LO0, LO1, . . . L08, L09

Note

OpenVMS network devices are always template devices and are enumerated as FWA0, FWB0, FWC0, . . . FWY0, FWZ0.

2.3.2 Specifying the Network Mask

An IP address consists of a network number and a host number. The network mask is the part of the host field of the IP address the identifies the network. Every host on the same network must have the same network mask. To specify the network mask, use the /NETWORK_MASK qualifier.

TCP/IP Services calculates the default by setting:

- The bits representing the network fields to 1
- The bits representing the host field to 0

You can also divide the host field into a site-specific network and host field.

2.3.3 Specifying Additional IP Addresses

To establish an additional IP address for an interface, define a network alias. This can be useful when changing network numbers and you want to continue to accept packets addressed to the old interface, or for setting up a host with a single interface to act as a router between subnets. Network aliases can be added in two functionally identical ways:

- Associate multiple addresses to an existing interface.

  You can use the `ifconfig` utility to associate multiple addresses with an existing interface. There is no limit to the number of aliases that can be created, and ranges of network addresses can be easily created. You should include the `ifconfig` command in SYS$STARTUP:TCPIP$SYSTARTUP.COM to ensure the network aliases are re-created whenever TCP/IP Services is restarted.

For example, assume interface WF0 exists with a network address of 10.10.1.100 and a 24-bit subnet mask. To add an alias with an address of 10.10.2.100 with a 24-bit subnet mask, follow these steps:

1. Define foreign commands:

```
$ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM
```

2. Display the current interfaces. Use quotation marks to preserve case. For example:

```
$ netstat -n "-I" wf0
Name  Mtu   Network     Address              Ipkts Ierrs    Opkts Oerrs  Coll
WF0   4470  <Link>      0:0:f8:bd:bc:22     3049700     0  2976912   0     0
WF0   4470  10.10.1     10.10.1.100         3049700     0  2976912   0     0
```

3. Add the network alias:

```
$ ifconfig wf0 alias 10.10.2.100/24
```

4. Display the current interfaces. For example:

```
$ netstat -n "-I" wf0
Name  Mtu   Network     Address              Ipkts Ierrs    Opkts Oerrs  Coll
WF0   4470  <Link>      0:0:f8:bd:bc:22     3049700     0  2976912   0     0
WF0   4470  10.10.1     10.10.1.100         3049700     0  2976912   0     0
WF0   4470  10.10.2     10.10.2.100         3049700     0  2976912   0     0
```

A range of network addresses can be associated with an interface by using the `aliaslist` parameter to the `ifconfig` command. For more information, enter the following command:

```
TCPIP> HELP IFCONFIG PARAMETERS
```

- Configure a pseudo-interface.

  A pseudo-interface can be created to associate another network address with the same physical interface also. Use the SET INTERFACE TCP/IP Services management command to create a pseudo-interface. See Section 4.4.3 for more information.

3
Configuring Serial Lines

A serial connection is made between two systems using modems and telephone lines or other serial lines. TCP/IP Services supports serial connections using the PPP (Point-to-Point Protocol) and SLIP (Serial Line IP) protocols. SLIP includes CSLIP (compressed SLIP). You can use any standard OpenVMS terminal device as a PPP or SLIP line. (PPP is available for OpenVMS Alpha systems only.)

This chapter reviews key concepts and describes:

- How to set up a PPP interface (Section 3.2)
- How to set up a SLIP interface (Section 3.3)
- How to solve serial line problems (Section 3.4)

3.1 Key Concepts

If your OpenVMS system is part of a large network, you will probably use both PPP and SLIP for your serial connections. As an Internet standard, PPP is often preferred because it ensures interoperability between systems from a wide variety of vendors. PPP provides a way for your OpenVMS Alpha system to establish a dynamic IP network connection over a serial line without an additional router or additional server hardware.

SLIP has been in use for a longer period of time and is available for most terminal servers and in most PC implementations of TCP/IP. Because SLIP and PPP do not communicate with each other, hosts wanting to communicate must use the same protocol. For example, if your terminal server supports only SLIP, remote hosts that connect through this server must also use SLIP.

3.1.1 PPP and SLIP

One of the largest applications for IP over serial lines is dialup access. With this type of configuration, the OpenVMS host answers calls and establishes a connection initiated by a user on a client host. The client host can be another OpenVMS system, a UNIX system, or a PC. Or users on the host can originate the dialup connection to a remote host or terminal server running the same protocol.

Dedicated serial lines running PPP or SLIP can also be used to connect separate LANs into a single WAN. In such a configuration, the host at each end of the serial connection is always the same; no other hosts are allowed to connect to either serial device.

3.1.2 Assigning an IP Address to Your PPP or SLIP Interface

Every network interface must have its own unique IP address. Interfaces cannot share IP addresses.

If you configure PPP interfaces for multiple remote hosts, the remote hosts can obtain their individual IP addresses from your host when they connect. Similarly, you can configure a PPP interface on your system without knowing your own IP address and obtain it when you connect to a remote system.

Before establishing SLIP communication with a remote host, however, you must obtain the IP address for the host's serial interface and assign IP addresses for each interface you configure on the local host.

When using SLIP, consider placing each serial line in a separate subnetwork. You accomplish this by assigning the same subnet mask for the interfaces at either end of the link.

If you need to use an address in the same subnetwork as your site LAN, use the proxy Address Resolution Protocol (ARP) feature (see Section 3.3.4).

3.1.3 Serial Line Internet Protocol

SLIP sends a datagram across the serial line as a series of bytes. It uses the following characters to determine when a series of bytes should be grouped together:

Character	Function	Hex Value	Decimal Values
END	Marks the end of the datagram. When the receiving SLIP encounters the END character, it knows that it has a complete datagram.	C0	192
ESC	Indicates the end of the SLIP control characters.	DB	219

The SLIP starts by sending an END character. If END is encountered within the datagram as data, the SLIP inserts an escape character, sending the two-character sequence DB DC instead. If the ESC character appears within the datagram as data, it is replaced with the two-character sequence DB DD. The datagram ends with the END character after the last byte in the packet is transmitted.

There is neither a standard SLIP specification nor a defined maximum packet size for the SLIP. The TCP/IP Services implementation of SLIP accepts 1006-byte datagrams and does not send more than 1006 bytes in a datagram.

Compressed SLIP provides header compression that is beneficial for small packets and low-speed serial links. Header compression improves packet throughput. You can enable the CSLIP by means of the /COMPRESS qualifier when you enter a SET INTERFACE command. See Table 3–3 for more information.

3.1.4 Point-to-Point Protocol

PPP uses a frame format that includes a protocol field. The protocol field identifies the protocol (for example, IP, DECnet, or OSI) to be used for communication between the two hosts. The PPP defines the network frame in a 5-byte header and 3-byte trailer. A PPP frame starts and ends with the control byte 7E hex (126 decimal). The address and control bytes are constant. The 2-byte protocol field indicates the contents of the PPP frame.

3.2 Setting Up a PPP Interface (Alpha Only)

Use the following commands to configure a PPP interface on an OpenVMS Alpha system:

- SET INTERFACE PP*n*, where *n* is the number of the interface, takes effect immediately and stays in effect until the next TCP/IP Services shutdown.
- SET CONFIGURATION INTERFACE PP*n*, where *n* is the number of the interface, makes the change part of the permanent configuration and takes effect at the next TCP/IP Services startup.

Note

Specifying PP without the interface number is equivalent to specifying PP0.

If you enter a SHOW INTERFACE command, the address does not appear until a PPP connection is actually established.

Table 3–1 shows the command qualifiers used for configuring PPP interfaces.

Table 3–1 Configuring PPP Interfaces

Qualifier	Description
/COMPRESS=[ON | OFF | AUTOMATIC]	Optional. The default is ON. Use to negotiate header compression.
/DESTINATION=[*host_name* | *IP_address*]	Optional. The default is no destination host. If you do not specify the client host's address, the PPP obtains the correct address from the client host. If the host is used as a dialup provider, use this command to specify a unique IP address for a client. In this case, you must also specify your host address with the /HOST qualifier.
/HOST=[*host_name* | *IP_address*]	Required when setting up a host as a dialup provider; otherwise optional. Host name or IP address using the interface. If your host is multihomed, specify the unique IP address if the two IP addresses map to the same host name.
/NETWORK_MASK=*IP_address*	Optional. The subnet mask of the local PPP interface in dotted-decimal notation.
/SERIAL_DEVICE=*device*	Required for hard-wired or dedicated modem connections. Identifies the OpenVMS device name assigned to the PPP interface, for example, TTA1.

3.2.1 Setting Up Your Host for PPP Connections

In the client/server model for PPP connections, a host can function as a server, or **dialup provider**, to respond to incoming PPP connection requests. A host can also function as a **client** dialing in to a dialup provider.

- A PPP dialup provider answers modem calls from PPP clients, assigns IP addresses, and establishes PPP connections initiated by client hosts.

  Typically, a PPP dialup provider is permanently connected to the network through an interface such as Ethernet. The dialup provider services PPP clients that initiate temporary, dialup connections because they do not have permanent connections.

- A PPP client establishes a temporary PPP connection to a dialup provider or a terminal server.

Note

For information about establishing a PPP client connection from a UNIX system, refer to the UNIX documentation. For a connection from a PC, refer to the PC's dialup networking instructions. You will need to configure your modem correctly as outlined in the Section 3.2.1.2.

Setting up an OpenVMS Alpha host as a PPP dialup provider or client involves a series of tasks. These tasks are listed in Table 3–2 in the order you should complete them, and are explained in Sections 3.2.1.1 through 3.2.1.6.

Table 3–2 Set Up Tasks Required for an OpenVMS Alpha PPP Dialup Provider or Client

Step	Task	OpenVMS Dialup Provider	OpenVMS Client
1	Install the correct terminal driver.	Yes	Yes
2	Configure your modem.	Yes	Yes
3	Set up an asynchronous port for modem connections.	Yes	Yes
4	Configure an interface for a serial PPP connection.	Yes	Optional
5	Enable IP forwarding and dynamic routing, as appropriate.	Yes	No
6	Initiate a PPP connection. NETMBX and OPER privileges required.	No	Yes

3.2.1.1 Installing the Terminal Driver

Confirm that the virtual terminal driver SYS$LOADABLE_IMAGES:SYS$TTDRIVER.EXE is installed on your host. If it is not installed, run the System Management utility (SYSMAN), connect the device, and load the driver, as shown in the following example:

```
$ RUN SYS$SYSTEM:SYSMAN

SYSMAN> IO CONNECT VTA0 /NOADAPTER /DRIVER=SYS$TTDRIVER
SYSMAN> EXIT
```

After you run SYSMAN, confirm that the VTA0 device was created. For more information about SYSMAN and its parameters, see the *OpenVMS System Management Utilities Reference Manual: M-Z*.

For OpenVMS Alpha Version 7.1, you must also install the ASNDRIVER remedial kit to prevent the system from crashing. To obtain the driver and associated corrections, access a remedial kit and accompanying cover letter from:

```
http://ftp.service.digital.com/public/vms/axp/v7.1/alppppd01_071.A-DCX_AXPEXE
http://ftp.service.digital.com/public/vms/axp/v7.1/alppppd01_071.CVRLET_TXT
```

3.2.1.2 Configuring the Modem

To configure the modem, follow these steps:

1. Make sure the serial port and modem cable support modem control signals. (Compaq's BC22F cable is an example of such a cable.)
2. Determine whether there are any baud rate restrictions associated with your phone line or on your connecting cable (when using a null modem or modem eliminator).
3. Adjust the settings on your modem to enable AT commands, as appropriate for your modem. Some modems require you to set DIP switches, while others require you to specify software settings.

   Sample DIP switch configuration settings for U.S. Robotics Courier modems are as follows. Note the following designations in these samples:

   X = setting on (although different settings might work)
   X** = setting on (required)

   Dialup provider settings:

```
DTR normal                X**     DTR always on
Verbal result codes       X       Numeric results codes
Suppress result codes     X**     Display result codes
Echo offline commands     X       No echo offline commands
Auto answer on ring       X**     Suppress auto answer
Normal carrier detect     X**     Carrier detect override
Display all results codes X       Result codes orig. mode only
Disable AT command set            Enable AT command set       X
Disconnect with +++               No disconnect with +++      X
Load NVRAM defaults       X       Load &FO settings
```

   Client settings (defaults):

```
DTR normal                X     DTR always on
Verbal result codes       X     Numeric results codes
Suppress result codes           Display result codes        X
Echo offline commands     X     No echo offline commands
Auto answer on ring             Suppress auto answer        X
Normal carrier detect     X     Carrier detect override
Display all results codes X     Result codes orig. mode only
Disable AT command set          Enable AT command set       X**
Disconnect with +++       X     No disconnect with +++
Load NVRAM defaults       X     Load &FO settings
```

4. If possible, also configure the modem so that it does not assert the Data Terminal Ready (DTR) signal until it asserts the Carrier Detect (CD) signal. This configuration ensures that the terminal driver does not drop the DTR signal prematurely.

3.2.1.3 Setting Up an Asynchronous Port

Use the DCL command SET TERMINAL and applicable qualifiers to set up an asynchronous port for use with the modem.

- Setting up the PPP dialup provider

  Enter the SET TERMINAL command and qualifiers appropriate for your modem connection. (Note that some qualifiers require LOG_IO or PHY_IO privilege, or both.) For example:

```
$ SET TERMINAL TTA0: /ALTYPEAHD /AUTOBAUD /DIALUP /DISCONNECT /EIGHTBIT -
_$ /MODEM /NOHANGUP /NOHOSTSYNC /NOPASTHRU /NOREADSYNCH /NOTTSYNCH -
_$ /PERMANENT /TYPE_AHEAD
```

  Where:

/ALTYPEAHD	Creates a permanent, alternate type-ahead buffer. (The system parameter TTY_ALTYPADH determines the size of the type-ahead buffer.) Helpful when transferring larger files. This qualifier is required.
/AUTOBAUD	Detects the incoming baud rate.
/DIALUP	Specifies that the terminal is a dialup terminal. This qualifier is required.
/DISCONNECT	Ensures that the process is disconnected if the line detects a hangup.
/EIGHTBIT	Sets the terminal to use the 8-bit ASCII format. This qualifier is required.
/MODEM	Specifies the use of a modem. This qualifier is required.
/NOHANGUP	Does not hang up the modem when the client logs off. This is the default. This qualifier is required.
/NOHOSTSYNC	Does not allow the use of Ctrl/S or Ctrl/Q functions from the terminal to stop or resume transmission when the input buffer is full or empty. This is the default.
/PASTHRU	The terminal passes format-type data, such as carriage returns and tabs, to an application program as binary data. This is the default.
/NOREADSYNCH	Does not allow the use of Ctrl/S or Ctrl/Q functions to synchronize data transmitted from the terminal. This is the default.
/NOTTSYNCH	Does not allow transmission to be stopped or resumed by entering Ctrl/S or Ctrl/Q, respectively.
/PERMANENT	Saves the settings.
/TYPE_AHEAD	Enables remote modems. Must be set. The terminal accepts unsolicited input to the limit of the type-ahead buffer. This is the default.

  For detailed information about these and other SET TERMINAL qualifiers, see the *OpenVMS DCL Dictionary: N-Z*.

- Setting up the PPP client (OpenVMS Alpha only)

  Enter the SET TERMINAL command and qualifiers appropriate for your connection, as listed for the dialup provider, with the exception of /AUTOBAUD.

  Set the baud rates using the /SPEED=(*input-rate,output-rate*) qualifier. If the rates are the same, specify /SPEED=*rate* (for example, /SPEED=9600).

3.2.1.4 Configuring a PPP Interface

- Configuring the PPP dialup provider

  Use the SET INTERFACE command and qualifiers to configure the interface for a serial PPP connection and assign a host name, IP address, network mask, and IP address for the client host, as applicable:

  ```
  TCPIP> SET INTERFACE PPn /SERIAL_DEVICE=TTn: /HOST=IP_address -
  _TCPIP> /NETWORK_MASK=IP_address /DESTINATION=IP_address /COMPRESS=AUTO
  ```

  In this command:

  – *n* is the controller name and unit number.

  – The /HOST address is the IP address.

  – The /NETWORK_MASK IP address is required if your network uses subnets.

  – The /DESTINATION address is the IP address assigned to the client host making a connection request. This address always overrides the client's own IP address, if the client has one.

  – /COMPRESS=AUTO turns off IP header compression unless the client uses it.

- Configuring the PPP client (OpenVMS Alpha only) (Optional)

  Use the SET INTERFACE command and /HOST qualifier to assign an IP address:

  ```
  TCPIP> SET INTERFACE PPn /SERIAL_DEVICE=TTn: /HOST=IP_address
  ```

  In this command, *n* is the interface number. If you omit the interface number, PP0 is used.

  If you do not specify your host's IP address using SET INTERFACE, the dialup provider or terminal server provides an IP address after the connection is established.

  Note

  If the connecting client host has only a loopback and tunnel interface defined:

  1. A default route to the PPP interface is added to the routing table when the connection is established.
  2. The IP address of the PPP interface is assigned to the logical names TCPIP$INET_HOSTADDR and UCX$INET_HOSTADDR (for backward compatibility).

3.2.1.5 Enabling IP Forwarding (Dialup Provider Only)

Enter the following command to enable IP forwarding:

```
TCPIP> SET PROTOCOL IP/FORWARD
```

To enable IP forwarding in the configuration database, enter the following command:

```
TCPIP> SET CONFIGURATION PROTOCOL IP/FORWARD
```

Alternatively, use the sysconfig utility. First, define the TCP/IP Services foreign commands:

```
$ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM
```

Enter the following SYSCONFIG commands:

```
$ SYSCONFIG -r inet ipforwarding=1
$ SYSCONFIG -r inet ipgateway=1
$ SYSCONFIG -q inet
```

To send notifications automatically on all connected LANs when new hosts or networks become reachable, use dynamic routing with the /SUPPLY option. For example, every time a PPP link is set up to a new subnetwork, RIP (Routing Information Protocol) advertises a corresponding route.

For example, enter the following commands:

```
TCPIP> START ROUTING /SUPPLY
TCPIP> SET CONFIGURATION START ROUTING /SUPPLY
```

If your PPP and Ethernet interfaces are in the same network, a route is created automatically for the client hosts and an ARP proxy is advertised.

3.2.1.6 Initiating a PPP Connection

You use the OpenVMS PPP utility (PPPD) and associated commands to establish and manage a temporary PPP connection from an OpenVMS Alpha client host to an OpenVMS dialup provider or terminal server. Note that NETMBX and OPER privileges are required to establish a successful connection and to display OPCOM messages.

To invoke PPPD, enter the DCL command PPPD. The PPPD commands are summarized in the following table. For detailed information about PPPD commands and qualifiers, enter the HELP command.

Command	Function
CONNECT	Establishes a network connection through the current physical port or a specified remote port.
DIAL_OUT	Allows direct access to a device in order to dial out over a modem or link to an external device.
DISCONNECT	Terminates the network connection and returns control to the terminal driver.
EXIT	Leaves the utility and returns you to the DCL command prompt ($).
HELP	Displays help text for PPPD commands.
SET	Determines the device and line characteristics for the specified terminal.
SHOW	Displays the device and line characteristics of the specified terminal.

To initiate a PPP connection from an OpenVMS Alpha client to an OpenVMS dialup provider or terminal server, follow these steps.

1. Confirm that you have NETMBX and OPER privileges.
2. Use the PPPD command DIAL_OUT and specify the terminal device. After the atdt command, enter the telephone number of the dialup provider or terminal server. (With some modems, you might need to type the number again until dialing begins.)

For example:

```
$ PPPD

PPPD> DIAL_OUT TTA0

Type control-~ to send a break
     control-\ to disconnect
     control-@ to switch to a Point-to-Point connection.

atdt 8671234
```

3. If you are connecting to another OpenVMS system, log in to the system after you dial up, and enter the following commands to establish the connection:

   ```
   $ PPPD
   PPPD> CONNECT
   ```

   To end the connection, enter the DISCONNECT TT*n* command at the PPPD> prompt and log out.

4. If you are connecting to a terminal server, enter the CONNECT PPP prompt at the LOCAL> prompt. An informational message will confirm the PPP connection:

   ```
   LOCAL> CONNECT PPP

   Local -561- Starting SLIP or PPP datalink session
   %PPPD-I-CONNECTTERM, converting connection on device _TTA0: to a
   Point-to-Point connection
   ```

   To end the connection, enter DISCONNECT TT*n* at the PPPD> prompt. After the connection is terminated, an OPCOM message is displayed. For example:

   ```
   %%%%%%%%%%  OPCOM   23-APR-1998 15:44:32.10  %%%%%%%%%%
   Message from user XYZnet on JONES
   %TCPIP-S-PPPDISCONN, Disconnected  PPP Interface PP1 on TTA0
   ```

3.2.2 Removing the PPP Configuration

To remove the PPP configuration, follow these steps:

1. If you created a PPP interface, return the associated terminal port to general use. Enter:

   ```
   TCPIP> SET NOINTERFACE PPn
   ```

   In this example, *n* is the number of the interface. If you omit the interface number, PP0 is assumed.

2. If you added special route and proxy entries with the PPP line, remove them.

3. If you changed any terminal settings in preparation for PPP, restore them. Enter the DCL command SET TERMINAL, and wait for the modem to reset and free the port and phone line.

3.3 Setting Up a SLIP Interface

Configuring the network interface for SLIP is the same as configuring the interface for Ethernet connections. In this case, the network interface is the modem connection. Remember that before you can configure a SLIP line, you must choose an IP address for the interface at each end of the line and establish a physical connection.

Use the following commands to set up the SLIP interface:

- SET INTERFACE SL*n*, where *n* is the number of the interface. If you omit the interface number, SL0 is assumed. This command takes effect immediately and stays in effect until the next TCP/IP Services shutdown.
- SET CONFIGURATION INTERFACE SL*n*, where *n* is the number of the interface. If you omit the interface number, SL0 is assumed. This command makes the change part of the permanent configuration. The change takes effect at the next product startup.

Table 3–3 describes the command qualifiers used for configuring SLIP interfaces.

Table 3–3 Command Qualifiers Used for Configuring SLIP

Qualifier	Description
/[NO]AUTO_START	Optional. The default is /AUTO_START. Automatically creates the interface on startup.
/COMPRESS=[ON | OFF | AUTOMATIC]	Optional. The default is no compression. Enables or disables TCP header compression (CSLIP). With /COMPRESS=AUTOMATIC, compression remains off unless the remote host begins to use it.
/[NO]FLOWCONTROL	Optional. The default is No flow control. Enables the special handling of XON and XOFF characters to work properly with modems that are configured to interpret these characters locally. Specify /FLOWCONTROL only if the host at the other end of the line is another host running TCP/IP Services. If you cannot use /FLOWCONTROL, configure your modem to pass all the XON and XOFF characters through transparently.
/HOST=(*host_name, IP_address*)	Required. Host name or IP address of the local host. If your host is multihomed, you must specify an address in dotted-decimal notation.
/NETWORK_MASK=*subnet_address*	Required. The subnet mask of the local SLIP interface in dotted-decimal notation.
/SERIAL_DEVICE=*device*	Required for hard-wired or dedicated modem connections. Optional for dynamic connections. Identifies the OpenVMS device name assigned to the SLIP interface, for example, TTA1.

For example, the following command configures SLIP interface SL5, using the local IP address assigned to host CROW, with a subnetwork mask of 255.255.255.0. The interface uses the terminal device TTA3:. The /COMPRESS qualifier enables TCP header compression (CSLIP). The /FLOWCONTROL qualifier enables special handling of XON and XOFF characters.

```
TCPIP> SET INTERFACE SL5 /HOST=CROW /NETWORK_MASK=255.255.255.0 -
_TCPIP> /SERIAL_DEVICE=TTA3 /COMPRESS=ON /FLOWCONTROL
```

3.3.1 Setting Up Hard-Wired SLIP Lines

To configure SLIP with hard-wired lines, follow these steps:

1. Establish a physical connection. Plug in a serial cable between the two host systems or ensure that they are both cabled to opposite ends of a leased line.
2. Obtain an IP address if necessary.
3. Configure the SLIP interface. Enter the SET INTERFACE command with the /HOST and /SERIAL_DEVICE qualifiers, which are required.

3.3.2 Setting Up SLIP Dialup Lines

You can configure either a terminal server port or an OpenVMS system to answer dialin calls.

Follow these steps:

1. Configure the appropriate settings for the terminal port to which you will connect. Begin a dialog of dialing (or answering) commands with your modem. The specific required commands depend on the type of modem you are using.

   For example, to prevent the modem from hanging up when you exit the DTE session to bring up the SLIP line, enter the following command:

   ```
   $ SET TERMINAL TTA2 /PERMANENT /MODEM /NOHANGUP
   ```

   To disable interactive logins on the line, enter the following command:

   ```
   $ SET TERMINAL TTA2 /PERMANENT /NOTYPEAHEAD
   ```

   Any SLIP data that arrives before you enter the SET INTERFACE command is ignored. Otherwise, this command triggers the creation of a new interactive login process.

   To enable interactive logins after a user sends a Break, enter the following command:

   ```
   $ SET TERMINAL TTA2 /PERMANENT /NOAUTOBAUD /SECURE_SERVER
   ```

2. Configure the modem. Enter the appropriate commands to dial the telephone and establish communication.
3. Unless you are setting up a SLIP line between two hosts running TCP/IP Services and plan to use the /FLOWCONTROL qualifier at both ends, disable modem recognition of XON and XOFF characters. (If SLIP packets have Ctrl/S and Ctrl/Q characters embedded in them as data, you must prevent the modem from trying to interpret these characters.)

   Either use hardware flow control or disable flow control entirely. The following examples disable all flow control.

   - With a DECmodem V32 in AT command mode, set the following values:
     - `AT%F0` — No speed buffering flow control
     - `AT%M0` — Disable speed buffering (optional)
   - With a DECmodem V32 in DMCL mode, set the following values:
     - `SET P2/SBU`
     - `SET P1/SBU`
     - *`prompts appropriate_answers`*

- With a U.S. Robotics Sportster modem, set the following values:
  - AT&B0 — Variable, follows connection rate (optional)
  - AT&H0 — Flow control disabled
  - AT&I0 — Software flow control disabled

4. Obtain IP addresses if necessary.
5. To dial in, follow these steps:
   a. Enter the SET HOST /DTE command:

      ```
      $ SET HOST /DTE TTnx
      ```

   b. Type the telephone number. For example:

      ```
      atdt telephone_number
      ```

   c. The connected system displays its interactive (command mode) prompt. You are talking to the terminal server and can now make the connection.

The following example shows a user named SLIP-USER at a PC named ROBIN with a 9600-baud modem, using terminal device TTA2 and connecting it to the port of a terminal server. In this example:

- The terminal server is a DECserver 700 terminal server.
- The user directs the modem to dial the telephone number 222-2222.
- The password prompt of the terminal server is #.
- The terminal server's current login password is hootowl.
- The terminal server's prompt is Local>.
- The user types Ctrl/\ (Ctrl key plus backslash) to escape from the terminal server to the SLIP host.
- The user defines interface SL2 and identifies it as SLIP device TTA1: with IP address 1.2.3.4. Communication on this line will use CSLIP.

```
$ SET HOST /DTE TTA2

%REM-I-TOQUIT, connection established
Press Ctrl/\ to quit, Ctrl/@ for command mode

atdt 2222222

CONNECT 9600

# hootowl (not echoed)

Network Access SW V1.5 for DS700-16
(c) Copyright 1994, Digital Equipment Corporation - All Rights Reserved
Please type HELP if you need assistance

Enter username>SLIP-USER

Local> CONNECT SLIP
Ctrl/\

TCPIP> SET INTERFACE SL2 /HOST=1.2.3.4 /NETWORK_MASK=255.255.255.0 -
_TCPIP> /SERIAL_DEVICE=TTA1: /COMPRESS=ON
```

3.3.3 Setting Up Your Host as a SLIP Dialup Provider

You can configure your host to answer calls and establish connections initiated by users on remote hosts.

To set up your host as a SLIP provider:

1. Over the line you will define as a SLIP line, dial in to the host.
2. Log in to the remote host.
3. Enter an appropriate SET INTERFACE command with the /SERIAL_DEVICE qualifier to turn the line into a SLIP line.

   For example, the following command creates a SLIP interface named SL5, using the terminal device associated with the session where the command is entered.

   ```
   TCPIP> SET INTERFACE SL5 /HOST=192.208.35.5 /SERIAL_DEVICE=TT
   ```

4. Log out.

As soon as you log out, your terminal port becomes a SLIP interface. Without causing the modem to hang up, start SLIP on the remote system.

To facilitate connection setup for end users, create a dedicated user name for each remote host that dials in. These users need to have a LOGIN.COM procedure that invokes appropriate SET TERMINAL commands and TCP/IP management SET INTERFACE commands, terminating with a LOGOUT command. Every user should specify a different SLIP interface name and host name (or IP address). These users require the OPER privilege to create interfaces.

You can enable IP forwarding on the SLIP provider host and start dynamic routing. For example, enter the following commands:

```
TCPIP> SET PROTOCOL IP /FORWARD
TCPIP> SET CONFIGURATION PROTOCOL IP /FORWARD
```

To send notifications automatically on all connected LANs when new hosts or networks become reachable, use dynamic routing with the /SUPPLY option. For example, every time a SLIP connection is set up to a new remote subnetwork, RIP (Routing Information Protocol) advertises a corresponding route. For example, enter the following commands:

```
TCPIP> START ROUTING /SUPPLY
TCPIP> SET CONFIGURATION START ROUTING /SUPPLY
```

3.3.4 Connecting a Host to the LAN

You can make your SLIP-connected host appear as if it were directly connected to the LAN. This is possible using a proxy ARP server (usually the same host that is acting as a SLIP gateway into the LAN).

To use proxy ARP (Address Resolution Protocol), assign to the remote host an IP address in the same subnetwork as the LAN. As other hosts on the LAN attempt to communicate with the remote host, the SLIP gateway answers ARP queries for the remote host by giving its own LAN address. The gateway then forwards packets across the SLIP line.

Many DECserver terminal server products support SLIP connections and implement proxy ARP. If you dial in from an OpenVMS host to a terminal server, the terminal server automatically detects your IP address and begins responding to ARP queries, forwarding packets as necessary.

To use proxy ARP with a DECserver terminal server, assign an IP address in the same subnetwork as the terminal server.

At the terminal server, enter the TCP/IP management command SHOW PORT SLIP. Verify that:

- An IP address has not already been associated with your port.
- Header compression is available, if you plan to use it.

3.3.5 Setting Up a SLIP Gateway with Proxy ARP

It is also possible to set up your host as a SLIP gateway with proxy ARP. You might prefer this approach if your dialin modems are attached directly to an OpenVMS system rather than to a terminal server.

Follow these steps on the host to become a SLIP gateway:

1. Create a SLIP interface in another network or subnetwork, for example:

   ```
   $ TCPIP SET INTERFACE SL0 /HOST=10.1.2.3 /SERIAL_DEVICE=TTA2
   ```

2. Add a host route for the remote system. For example:

   ```
   $ TCPIP SET ROUTE FINCH /GATEWAY=10.1.2.3
   ```

3. Configure an ARP entry for the remote host, listing your own Ethernet address (as shown in TCPIP SHOW INTERFACE /FULL). For example:

   ```
   $ TCPIP SET ARP 08-00-2B-2C-4F-46 FINCH /PUBLIC
   ```

4. Enable IP packet forwarding, if not already done. Enter:

   ```
   $ TCPIP SET PROTOCOL IP /FORWARD
   ```

When your host is set up as a SLIP gateway, create an interface on the remote host at the other end of the serial line. Specify an address in the same subnetwork as the LAN.

Although the two ends of the SLIP line are in different subnetworks, traffic can flow properly due to the interface route you added with the SET ROUTE command.

3.3.6 Shutting Down SLIP

To terminate a SLIP connection, follow these steps:

1. Return the associated terminal port to general use. Enter:

   ```
   $ TCPIP SET NOINTERFACE interface
   ```

2. If you added special route and proxy entries in conjunction with the SLIP line, remove them.
3. If you changed any terminal settings in preparation for SLIP, restore them using the SET TERMINAL command.

3.4 Solving Serial Line Problems

If you have problems dialing in to an OpenVMS system using SLIP or PPP after following the instructions in this chapter, perform the following steps to isolate the cause of the problem:

1. Check the equipment used by both the client and the dialin provider:
   - Do the cables work?

- Are the modems configured properly?
- Are the DIP switches on the modems set correctly?
- Are the modem software settings correct? Make sure that flow control is disabled.
- Are all clients and dialup providers using unique addresses?

After a software upgrade, be sure to reboot and restart TCP/IP Services.

2. Make sure the SET HOST attempts have not exceeded the OpenVMS security level. To check and then delete, if necessary, any information about these attempts, enter the following commands. Note that SECURITY privilege must be enabled to use these commands.

```
$ SHOW INTRUSION
$ DELETE/INTRUSION_RECORD source
```

3. Make sure that IP forwarding is enabled using the following command:

```
TCPIP> SHOW PROTOCOL IP/FORWARD
```

4. Make sure the terminal characteristics for the terminal device associated with the interface are set up as follows:

```
$ SET TERMINAL TTnx /ALTYPEAHD /AUTOBAUD /DIALUP -
_$ /DISCONNECT /EIGHTBIT /MODEM /NOHANGUP /NOHOSTSYNC /NOPASTHRU -
_$ /NOREADSYNCH /NOTTYSYNCH /PERMANENT /TYPE_AHEAD
```

   Make sure you specify the /TYPE_AHEAD qualifier when you enter the SET TERMINAL command to set up an asynchronous port.

5. Enter the SET HOST/DTE command to make sure you can log in to the system:

```
$ SET HOST/DTE TTnx
```

   If you cannot log in to or communicate with the system, you may be using the wrong terminal device name (TT*nx*).

6. Set up OPCOM to receive messages using the DCL command REPLY/ENABLE. You need OPER privileges to use OPCOM.

7. You need NETMBX and OPER privileges to establish a successful connection. If these privileges are not enabled when you enter the CONNECT command, you will see messages similar to the following:

```
$ PPPD
PPPD> CONNECT

\}‘}"}(}"6~ <CTRL/@>
%PPPD-I-CONNECTTERM, converting connection on device _TTA0: to a
Point-to-Point connection
%PPPD-E-CALLBACKERR, error calling network callback
%SYSTEM-F-NOPRIV, insufficient privilege or object protection violation
%PPPD-F-ABORT, fatal error encountered; operation terminated
```

   Note that the extraneous data in this sample is an ASCII representation of IP packets transmitted over the open line.

   PPP sets up a default route on the client if one did not exist. Typically, a default route exists if another interface exists on the client.

8. Attempt to ping the remote system:

```
TCPIP> PING host-name
```

Watch the modem's LED display as you attempt to communicate using the PING command.

You might not be able to ping the system if the serial line is tied up with a large FTP operation.

9. Use the TCPTRACE command to see packets going in and out of the local system. For information about using TCPTRACE, enter:

```
$ HELP TCPTRACE
```

10. Display a count of the packets being sent and received on the problem interface, in full screen format, updated every second. For a SLIP problem, enter:

```
TCPIP> SHOW INTERFACE SLn
```

To display the packet counts for PPP problem, enter:

```
TCPIP> SHOW INTERFACE PPn
```

In these commands, *n* is the interface number.

3.4.1 Solving PPP Problems

Keep the following in mind for PPP-specific problems:

- If the virtual terminal software has not been loaded, the following error will be displayed when you try to connect:

```
%PPPD-E-NEEDVIRTTERM, point-to-point connection on device _TTB0: must
be done on a virtual terminal
```

Correct this problem by entering the following commands before you dial out:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> IO
CONNECT VT/NOADAPTER/DRIVER=SYS$LOADABLE_IMAGES:SYS$TTDRIVER.EXE
SYSMAN> EXIT
```

To make this permanent, add the following commands to the SYS$MANAGER:SYSTARTUP_VMS.COM file:

```
$ SET PROCESS/PRIVILEGE=CMKRNL
$ SYSMANIO = "SYSMAN IO"
$ SYSMANIO CONNECT VT/NOADAPTER/DRIVER=SYS$LOADABLE_IMAGES:SYS$TTDRIVER.EXE
```

Be sure to terminate any old virtual terminal sessions.

- If you are trying to use OpenVMS as a PPP client to your ISP (Internet service provider), check where the ISP uses an authentication protocol, such as CHAP, PAP, or RADIUS. These protocols are not supported and will prevent a connection to OpenVMS.

4
Configuring Routing

Routing allows traffic from your local network to reach its destination elsewhere on the internet. Hosts and gateways on a network use routing protocols to exchange and store routing information. Routing is the act of forwarding datagrams based on information stored in a routing table.

The TCP/IP Services product provides two types of routing: static and dynamic. This chapter reviews key routing concepts and describes:

- How to configure static routes (Section 4.2)
- How to enable and disable dynamic routing (Section 4.3)
- How to configure GATED (Section 4.4)

4.1 Key Concepts

If the hosts on your network need to communicate with computers on other networks, a route through a gateway must be defined. All hosts and gateways on a network store information about routes in routing tables. With TCP/IP Services, routing tables are maintained in both dynamic and permanent memory.

You can define routes manually (static routing), or you can enable routing protocols that exchange information and build routing tables based on the information exchanged (dynamic routing).

4.1.1 Static Routing

Because static routing requires manual configuration, it is most useful when the number of gateways is limited and where routes do not change frequently. For information on manually configuring routing, see Section 4.2.

4.1.2 Dynamic Routing

Complex environments require a more flexible approach to routing than a static routing table provides. Routing protocols distribute information that reflect changing network conditions and update the routing table accordingly. Routing protocols can switch to a backup route when a primary route becomes unavailable and can determine the best route to a given destination.

Dynamic routing tables use information received by means of routing protocol updates; when routes change, the routing protocol provides information about the changes.

Routing daemons implement a routing policy, that is, the set of rules that specify which routes go into the routing table. A routing daemon writes routing messages to a routing socket, causing the kernel to add a new route, delete an existing route, or modify an existing route.

The kernel also generates routing messages that can be read by any routing socket when events occur that may be of interest to the process, for example, the interface has gone down or a redirect has been received.

TCP/IP Services implements two routing daemons: the Routing Daemon (ROUTED) and the Gateway Routing Daemon (GATED). The following sections provide more information.

4.1.2.1 Routing Daemon (ROUTED)

This daemon (pronounced route-dee) supports the Routing Information Protocol (RIP). When ROUTED starts, it issues routing update requests then listens for responses. A system configured to supply RIP information responds to the request with an update packet. The update packet contains destination addresses and routing metrics associated with each destination. After receiving a RIP update, the ROUTED uses the information to update its routing table.

To configure dynamic routing with ROUTED, see Section 4.3.

4.1.2.2 Gateway Routing Daemon (GATED)

This daemon (pronounced gate-de) supports interior and exterior gateway protocols. It obtains information from several routing protocols and selects the best routes based on that information. You can configure GATED to use one or more of the protocols described in Table 4–1.

Table 4–1 GATED Routing Protocols

Protocol	RFC	Description
Routing Information Protocol (RIP) Versions 1 and 2	RFC 1058, RFC 1723	RIP is a commonly used interior protocol that selects the route with the lowest metric (hop count) as the best route.
Open Shortest Path First (OSPF) Version 2	RFC 1583	Another interior routing protocol, OSPF is a link-state protocol (shortest path first) and better suited than RIP for use in complex networks with many routers.
Exterior Gateway Protocol (EGP)	RFC 904	EGP exchanges reachability information between autonomous systems. An autonomous system is usually defined as a set of routers under a single administration, using an interior gateway protocol and common metric to route packets. Autonomous systems use exterior routing protocols to route packets to other autonomous systems.
Border Gateway Protocol (BGP)	RFCs 1163, 1267, 1771	Like EGP, BGP exchanges reachability information between autonomous systems but supports nonhierarchical topologies. BGP uses path attributes to provide more information about each route. Path attributes can include, for example, administrative preferences based on political, organizational, or security considerations.
Router Discovery	RFC 1256	This protocol is used to inform hosts of the availability of routers that it can send packets to, and to supplement a statically configured default router.

These routing protocols are configured in the GATED configuration file TCPIP$GATED.CONF. This file contains statements that control tracing options, select routing protocols, manage routing information, and manage independent system routing.

For information on configuring dynamic routing with GATED, see Section 4.4.

4.2 Configuring Static Routes

The first time you run the configuration procedure, TCPIP$CONFIG.COM, static routing is configured automatically. To manually configure static routing, use the CREATE ROUTE command to create an empty routes database file.

The default file name is SYS$COMMON:[SYSEXE]TCPIP$ROUTE.DAT. To specify a different name, define the systemwide logical name TCPIP$ROUTE.

Note

Do not enter the CREATE ROUTE command unless you intend to reconfigure your entire cluster.

4.2.1 Creating a Default Route

When TCP/IP is sending a packet, it consults the routing table to determine which interface is connected to the destination network. If the packet has a destination network address that is unknown, the packet is sent to the default router. The default route points at the default router. For example, if a router with address 16.20.0.173 is designated to route all packets between the local network and the rest of the world, then the default route can be set with the following command:

```
$ TCPIP SET ROUTE /DEFAULT /GATEWAY=16.20.0.173
```

If TCP/IP Services is active, this affects the active routes database. To ensure this default route is available next time TCP/IP Services is started, the /PERMANENT qualifier must be used. For example:

```
$ TCPIP SET ROUTE /DEFAULT /GATEWAY=16.20.0.173 /PERMANENT
```

Use the SET NOROUTE command to remove a route.

Or you can define the default route using the `route` UNIX command. In this case, to ensure the default route is recreated next time TCP/IP Services is started, add the command to SYS$STARTUP:TCPIP$SYSTARTUP.COM. For example, to create the same default route as defined above, use the following UNIX style command:

```
$ route add default 16.20.0.173
```

To remove the route, enter the following command:

```
$ route delete default 16.20.0.173
```

4.2.2 Manually Defining Static Routes

To create a static route, use the SET ROUTE command. The command has the following effects:

- If TCP/IP Services is not active, SET ROUTE modifies the permanent database.
- If TCP/IP Services is active, SET ROUTE modifies the volatile database.
- If TCP/IP Services is active, SET ROUTE/PERMANENT updates the permanent database.

The SET ROUTE command requires the following information:

- The IP address or domain name of the destination host or network

- The IP address or host name of a gateway that can reach the destination host

Compaq strongly recommends that you do not specify alias names with the *destination* parameter or the /GATEWAY=*host* qualifier.

To define a route to any host on a specific network, enter:

```
TCPIP> SET ROUTE network_IP_address /GATEWAY="gateway" /NETWORK
```

To define a route to a specific host on a specific network, enter:

```
TCPIP> SET ROUTE remote_host /GATEWAY="gateway"
```

4.2.2.1 Examples

1. In the following example, the network is active. The SET ROUTE command adds a route to the volatile routes database. TCPIP starts directing communication for flamingo through gateway francolin.

   ```
   TCPIP> SET ROUTE "flamingo" /GATEWAY="francolin"
   ```

2. In the following example, the network is active. The SET ROUTE command defines a routing path in the volatile routes database. The command specifies that traffic for the network with IP address 128.30.0.0 uses gateway francolin.

   ```
   TCPIP> SET ROUTE 128.30.0.0 /NETWORK /GATEWAY="francolin"
   ```

3. In the following example, the network is not active. The SET ROUTE command adds the new route to the permanent routes database. The next time the product starts up, packets for NENE will go through a gateway called bird.of.paradise.

   ```
   TCPIP> SET ROUTE NENE /GATEWAY="bird.of.paradise"
   ```

   At startup, the information in the permanent routes database, if any exists, is loaded into the volatile routes database. You can add permanent routes while the product is stopped or while it is running. If it is running, use the /PERMANENT qualifier.

4. The following command permanently sets routing for host albatross to go through gateway birdygate.

   ```
   TCPIP> SET ROUTE "albatross" /GATEWAY="birdygate" /PERMANENT
   ```

   A default route is a route used to direct data that is addressed to an unidentifiable network address. To define a default route, use the /DEFAULT qualifier.

5. The following command sets a default route. NIGHTINGALE is the default gateway.

   ```
   TCPIP> SET ROUTE /DEFAULT /GATEWAY=NIGHTINGALE
   ```

   To check that your routes are set up correctly, use either the LOOP or PING command.

4.2.3 Displaying Manually Defined Routes

To display static routes, use the SHOW ROUTE command. To see the permanent database, specify the /PERMANENT qualifier.

The display shows the following types of routes:

- A — Active route (A route that was created manually or associated with an interface.)
- D — Dynamic route. (A route that was dynamically created by the ROUTED or GATED routing daemon.)
- H — Host route (A route to a host.)
- N — Network route (A route to a network.)
- P — Permanent route (A route from the route database.)

To display a route that was defined by an address, specify either its address or a wildcard.

Some examples of displaying routes are listed below.

1. The following example displays information about all the manually defined routes.

```
TCPIP> SHOW ROUTE /FULL
                                DYNAMIC
Type           Destination              Gateway

AN   11.111.0.0   destin_host1   11.110.5.118   gate_host
AH   22.111.4.10  destin_host2   22.110.5.120   gate_host_2
```

2. The following example displays the permanent static routes that were defined with SET ROUTE/PERMANENT.

```
TCPIP> SHOW ROUTE /PERMANENT
                                PERMANENT
Type           Destination              Gateway

PN    0.0.0.0                    11.20.208.100   pterodactyl.extinct.com
PN    1.1.1.1                    22.2.2.2
```

Another way to display route information is by using the `netstat` UNIX command. For example, to display the routes, and suppress conversion of network numbers to names, enter the following commands:

```
$ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM

$ netstat -rn
Routing tables
Destination      Gateway            Flags     Refs     Use  Interface

Route Tree for Protocol Family 26:

Route Tree for Protocol Family 2:
default          16.20.0.173        UG          0        0  WE1
default          16.20.0.173        UG          0        0  WE0
16.20/16         16.20.208.161      U           2       56  WE1
16.20/16         16.20.208.160      U           1        9  WE0
16.20.208.160    16.20.208.160      UHL         0        0  WE0
16.20.208.161    16.20.208.161      UHL         0        0  WE1
127.0.0.1        127.0.0.1          UHL         1        1  LO0
```

This example shows a multihomed host with two interface adapters. For more information about the netstat utility, enter the following command:

```
TCPIP> HELP NETSTAT
```

4.3 Enabling and Disabling Dynamic Routing

Use the configuration procedure TCPIP$CONFIG to enable dynamic routing and configure your host to receive routing protocol messages as follows:

1. Select the Routing option from the Core Environment menu.
2. Answer “Yes” to the question "Do you want to configure dynamic ROUTED or GATED routing [NO]:"
3. You are asked whether you want to enable GATED.

   ```
   Do you want to enable GATED routing configuration?
   ```

   If you answer “Yes” to this question, GATED will be enabled. If you answer “No,” ROUTED will be enabled.
4. If you choose to enable ROUTED, indicate whether you want your host to supply RIP updates to other hosts on the network (in addition to receiving RIP updates) and the default network route.
5. If you choose to enable GATED, you must also configure the routing protocols in the GATED configuration file TCPIP$GATED.CONF. See Section 4.4 for more information about configuring GATED.

To disable dynamic routing:

1. Select the “Routing” option from the CORE ENVIRONMENT menu.
2. Answer “Yes” to the following questions:

   ```
   Do you want to reconfigure dynamic ROUTED or GATED routing [NO]: Y
   ```

   ```
   Do you want to disable dynamic ROUTED or GATED routing configuration
   [NO]: Y
   ```

   Alternatively, enter the TCP/IP management command STOP ROUTING.

When you disable GATED routing, the GATED routes are preserved. To disable GATED and remove all GATED routes from the routing table, enter the command STOP ROUTING/GATED.

4.4 Configuring GATED

You must configure the GATED protocols before starting GATED routing. Edit a copy of the sample file TCPIP$GATED.TEMPLATE (located in the SYS$SYSDEVICE:[TCPIP$GATED] directory) to add statements that select routing protocols, manage routing information, manage independent system routing, and control tracing options.

1. Use TCPIP$CONFIG to enable GATED.
2. Edit the TCPIP$GATED.TEMPLATE file.
3. Save the file TCPIP$GATED.CONF in the SYS$SYSDEVICE:[TCPIP$GATED] directory.
4. If GATED is already running, stop it by entering the command STOP ROUTING/GATED.

5. Start GATED by entering the command START ROUTING/GATED.

   See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for detailed descriptions of the SET GATED and START ROUTING/GATED commands.

If you do not format the configuration file correctly, GATED terminates.

For specific information about how to edit the GATED configuration file, see Appendix A.

4.4.1 Datagram Reassembly Time

Reassembly is the process of reconstructing a complete data message from received fragments. The reassembly timer determines the length of time allowed for the reassembly process. You can modify the reassembly timer to ensure that IP datagram fragments are optimally reassembled at the destination host.

Consider the following when setting the reassembly timer:

- If the timer expires before the host receives all the fragments, they are discarded.
- An inappropriately small interval might result in too many datagrams being discarded.
- An excessive interval might degrade internet performance.

Enter the following commands to reset the reassembly timer:

- For the current session:

  ```
  TCPIP> SET PROTOCOL IP /REASSEMBLY_TIMER=n
  ```

- To reset at the next TCP/IP Services startup:

  ```
  TCPIP> SET CONFIGURATION PROTOCOL IP /REASSEMBLY_TIMER=n
  ```

In the following example, the first command changes the IP reassembly time to 20 seconds on the running system. This new setting remains in effect until the next TCP/IP Services startup.

The second command makes the change permanent by modifying the configuration database, TCPIP$CONFIGURATION.DAT.

```
TCPIP> SET PROTOCOL IP /REASSEMBLY_TIMER=20
TCPIP> SET CONFIGURATION PROTOCOL IP /REASSEMBLY_TIMER=20
```

4.4.2 Enabling Forwarding

To enable packet forwarding between networks, enter the following TCP/IP management command:

```
TCPIP> SET PROTOCOL IP /FORWARD
```

To ensure this is set up the next time TCP/IP Services is restarted, enter the following command:

```
TCPIP> SET CONFIGURATION PROTOCOL IP /FORWARD
```

Display the setting using the following command:

```
TCPIP> SHOW PROTOCOL /PARAMETERS
```

Or use the sysconfig utility to enable forwarding. First, define foreign commands:

```
$ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM
```

Enter the following sysconfig command:

```
$ sysconfig -r inet ipforwarding=1 ipgateway=1
```

To make sure forwarding is enabled after restarting TCP/IP Services, add the command to TCPIP$SYSTARTUP.COM.

To view the setting, use the following command:

```
$ sysconfig -q inet ipforwarding ipgateway
```

When multiple networks share the same physical media and the host has just one interface, it is still possible to forward packets between these networks by creating a network alias, as described in Section 2.3.3.

For example, consider a network in which two networks have network addresses of 16.20.1/24 and 16.20.2/24, and the host address is 180. If the host has a single ethernet interface, WE0, create the interface and pseudointerfaces as follows:

```
TCPIP> SET CONFIGURATION INTERFACE WE0 /HOST=16.20.1.180 -
_TCPIP> /NETWORK_MASK=255.255.255.0 /BROADCAST_MASK=16.20.1.255

TCPIP> SET CONFIGURATION INTERFACE WEA0 /HOST=16.20.2.180 -
_TCPIP> /NETWORK_MASK=255.255.255.0 /BROADCAST_MASK=16.20.2.255

TCPIP> SET CONFIGURATION PROTOCOL IP /FORWARD
```

When TCP/IP Services is restarted, the host will forward packets between these networks.

Alternatively, you can add the following commands to TCPIP$SYSTARTUP.COM and then restart TCP/IP Services:

```
$ ifconfig we0 aliaslist 16.20.1-2.180/24

$ sysconfig -r inet ipforwarding=1 ipgateway=1
```

4.4.3 Extending Routing

To use extended routing, define pseudointerfaces. A **pseudointerface** is a data structure that extends routing. Like an interface, the name of an internet pseudointerface is three alphabetic characters, followed by the pseudointerface unit number in the range of 0 through 255.

The first two characters are the same as the two characters in the internet interface name (interface type and interface class). See Section 2.3.1 for more information about interface names.

The third character identifies the controller letter that corresponds to the OpenVMS hardware controller.

For example, for an OpenVMS Alpha system with two Ethernet controllers, EZA0 and EZB0, you can define the following internet interfaces and pseudointerfaces:

- Internet interfaces:
  - ZE0
  - ZE1

- Internet pseudointerfaces, each with its own IP address, network mask, and broadcast mask:
  - SEA
  - SEA0
  - SEA1

    .
    .
    .

  - SEA254
  - SEB255

To extend routing, follow these steps:

1. Define the pseudointerfaces using the SET INTERFACE and SET CONFIGURATION INTERFACE commands:

   ```
   TCPIP> SET NOINTERFACE interface

   TCPIP> SET INTERFACE interface /HOST=host -
   _TCPIP> /NETWORK_MASK=mask /BROADCAST_MASK=b_mask

   TCPIP> SET CONFIGURATION INTERFACE interface /HOST=host -
   _TCPIP> /NETWORK_MASK=mask /BROADCAST_MASK=b_mask
   ```

   For example, to specify the pseudointerface FFA0 on host KESTREL, with network mask 255.255.0.0 and broadcast mask to 128.30.0.0, enter:

   ```
   TCPIP> SET NOINTERFACE FFA0

   TCPIP> SET INTERFACE FFA0 /HOST=KESTREL /NETWORK_MASK=255.255.0.0 -
   _TCPIP> /BROADCAST_MASK=128.30.0.0
   ```

2. Enter the same information into the configuration database to set up the interfaces at startup. For example:

   ```
   TCPIP> SET CONFIGURATION INTERFACE FFA0 /HOST=KESTREL -
   _TCPIP> /NETWORK_MASK=255.255.0.0 /BROADCAST_MASK=128.30.0.0
   ```

   To display information about the network interfaces, use the SHOW INTERFACE command. To remove the interface from the configuration database, use the SET CONFIGURATION NOINTERFACE command.

4.4.4 Interface Routes

If you have a configuration in which multiple networks share the same physical LAN, you can communicate directly with hosts in other networks without the need of a pseudointerface for each network.

You can use a broadcast address to designate an interface route, also called a metric 0 route.

To create interface routes, follow these steps:

1. As the gateway for the route, enter either one of the host's own addresses or the broadcast address associated with an interface.

   TCP/IP Services recognizes this route as an interface route.

2. Configure the hosts in the other network to recognize that your network is present on their LAN.

For example, network 99.0.0.0 is on the same cable as network 192.199.199.0. On host 99.1.2.3, specify network 192.199.199.0 as directly reachable:

```
TCPIP> SET ROUTE 192.199.199.0 /NETWORK /GATEWAY=99.1.2.3
```

On the hosts in network 192.199.199.0, enter:

```
TCPIP> SET ROUTE 99.0.0.0 /NETWORK /GATEWAY=192.199.199.255
```

4.4.5 Manually Configuring a Hardware Address

Network hosts require manual configuration of a hardware address for a remote IP address under the following conditions:

- The remote host does not support the Address Resolution Protocol (ARP). You need static mapping of IP addresses to hardware addresses.
- The remote host is running ARP, but a change was made to the internet interface on that host.

  To notify your system about the change, flush the address mapping tables. Use the SET NOARP command to do this.

For example, to map the Ethernet address AA-02-04-05-06-07 of host ROOK, add the hardware address to the ARP table by entering the following command:

```
TCPIP> SET ARP AA-02-04-05-06-07 ROOK
```

Part 2
BIND

Part 2 provides information on configuring and managing the TCP/IP Services name server and includes the following chapters:

- Chapter 5, Configuring and Managing BIND, describes how to configure and manage the TCP/IP Services implementation of the Berkeley Internet Name Domain (BIND) software.
- Chapter 6, Using DNS to Balance Work Load, describes how to use BIND's round-robin scheduling or the load broker for cluster load balancing.

5
Configuring and Managing BIND

The Domain Name System (DNS) is a system that maintains and distributes information about Internet hosts. DNS consists of several databases that store host names and host IP addresses. With DNS, there is no central storage of data — no one server knows everything about all the Internet domains.

In UNIX environments, DNS is implemented by the Berkeley Internet Name Domain (BIND) software. Compaq TCP/IP Services for OpenVMS implements a BIND server based on the Internet Software Consortium's (ISC) BIND 8.1.2.

BIND 8.1.2 provides the following features:

- DNS dynamic updates
- DNS change notification
- New configuration syntax
- Flexible, categorized logging system
- IP address-based access control for queries, zone transfers, and updates that can be specified on a zone-by-zone basis
- More efficient zone transfers
- Improved performance for servers with thousands of zones

This BIND implementation also includes round-robin scheduling. A more robust load-balancing mechanism is provided with the load broker, which uses standard DNS dynamic updates.

This chapter contains the following topics:

- A review of key BIND concepts (Section 5.1)
- How to migrate your existing BIND environment to BIND 8 (Section 5.2)
- How to configure BIND using the BIND configuration file (Section 5.3)
- How to configure dynamic updates (Section 5.3.6)
- How to configure a DNS cluster failover and redundancy environment (Section 5.3.7)
- How to populate the BIND server databases (Section 5.4)
- How to configure BIND using SET CONFIGURATION BIND commands (Section 5.6)
- How to configure the BIND resolver (Section 5.7)
- How to use NSLOOKUP to query a name server (Section 5.8)
- How to troubleshoot BIND server problems (Section 5.9)

5.1 Key Concepts

This section serves as a review only and assumes you are acquainted with the InterNIC, that you applied for an IP address, and that you registered your domain name. You should also be familiar with BIND terminology, and you should have completed your preconfiguration planning before using this chapter to configure and manage the BIND software.

If you are not familiar with DNS and BIND, see the *DIGITAL TCP/IP Services for OpenVMS Concepts and Planning* guide. If you need more in-depth knowledge, see O'Reilly's *DNS and BIND, Third Edition*. For details on BIND 8 configuration, see the ISC BIND 8 documentation at `http://www.isc.org/`.

5.1.1 How the Resolver and Name Server Work Together

BIND is conceptually divided into two components: a resolver and a name server. The resolver is software that queries a name server; the name server is the software process that responds to a resolver query.

Under BIND, all computers use resolver code but not all computers run the name server process.

The BIND name server runs as a distinct process called TCPIP$BIND. On UNIX systems, the name server is called `named` (pronounced name-dee). Name servers are typically classified as master (previously called primary), slave (previously called secondary), and caching-only servers, depending on their configurations.

5.1.2 Common BIND Configurations

You can configure BIND in several different ways. The most common configurations are resolver-only systems, master servers, slave servers, forwarder servers, and caching-only servers. A server can be any of these configurations or can combine elements of these configurations.

A group of database files containing BIND statements and resource records are used by servers. These files include:

- The forward translation file, *domain_name*.DB

  This file maps host names to IP addresses.

- The reverse translation file, *address*.DB

  This file maps the address back to the host names. This address name lookup is called reverse mapping. Each domain has its own reverse mapping file.

- Local loopback forward and reverse translation files, LOCALHOST.DB and 127_0_0.DB

  These local host databases provide forward and inverse translation for the widely used LOCALHOST name. The LOCALHOST name is always associated with the IP address 127.0.0.1 and is used for loopback traffic.

- The hint file, ROOT.HINT

  This file contains the list of root name servers.

A configuration file, TCPIP$BIND.CONF, contains statements that pull all of the database files together and governs the behavior of the BIND server.

5.1.2.1 Master Servers

A master server is the server from which all data about a domain is derived. Master servers are **authoritative**, meaning they have complete information about their domain and their responses are always accurate.

To provide central control of host name information, the master server loads the domain's information directly from a disk file created by the domain administrator. When a new system is added to the network, only the database located on the master server needs to be modified.

A master server requires a complete set of configuration files: zone, reverse domain, configuration file, hint file, and loopback files.

5.1.2.2 Slave Servers

Slave servers receive authority and their database from the master server.

A particular domain's database file is called a zone file and copying this file to a slave server is called a zone file transfer. A slave server assures that it has current information about a domain by periodically transferring the domain's zone file. Slave servers are also authoritative for their domain.

Configuring a slave server is similar to configuring a master server. The only difference is that you need to provide the name of the master server from which to transfer zone data.

Note

If you create a master, slave, or forwarder server for the same domain on which your local host resides, you should reconfigure your BIND resolver so that it uses this system (LOCALHOST) as its name server.

Slave servers require a configuration file, a hint file, and loopback files.

5.1.2.3 Caching-Only Servers

Caching-only servers get the answers to all name service queries from other name servers. Once a caching server receives an answer to a query, it saves the information and uses it in the future to answer queries itself. Most name servers cache answers and use them in this way but a caching-only server depends on this for all its server information. It does not keep name server database files as other servers do. Caching-only servers are **nonauthoritative**, meaning that their information is secondhand and can be incomplete.

Caching-only servers require a hint file and loopback files.

5.1.2.4 Forwarder Servers

The forwarding facility can be used to create a large sitewide cache on a few servers, reducing traffic over links to external name servers. Forwarder servers process requests that slave servers cannot resolve locally (for example, because they do not have access to the Internet).

Forwarding occurs on only those queries for which the server is not authoritative and does not have the answer in its cache.

A master or slave server specifies a particular host to which requests outside the local zone are sent. This is a form of Internet courtesy so that only a limited number of hosts actually communicate with the root servers listed in the file ROOT.HINT.

If you configure a forwarder server, you must provide the name of the host to which requests outside your zones of authority are forwarded.

5.2 Migrating to BIND 8.1

If you set up your BIND environment using a previous version of the TCP/IP Services product, you must convert the UCX databases and configuration information to the new BIND 8.1 format.

To convert your BIND configuration, enter the following command:

```
TCPIP> CONVERT/CONFIGURATION BIND
```

This command extracts the BIND-specific configuration information from UCX$CONFIGURATION.DAT and creates the BIND 8.1 configuration file TCPIP$BIND.CONF. It renames your BIND databases, where necessary (see Section 5.2.1 for more information).

You can continue to use the SET CONFIGURATION BIND commands to make changes to your configuration (see Section 5.6), or you can make changes by editing the text file TCPIP$BIND.CONF (see Section 5.3). If you continue to use the SET CONFIGURATION BIND commands, you must also enter the CONVERT/CONFIGURATION BIND command in order for your changes to take effect.

5.2.1 Navigating Two Different BIND Environments

This section summarizes the differences between the UCX BIND implementation and BIND 8.1.

It is important to remember that in BIND 8.1, name servers are configured by editing a text configuration file. The use of this file is described in Section 5.3. Compaq recommends, but does not require, that you use the configuration file to set up BIND. You can continue using the TCPIP$CONFIG and the SET CONFIGURATION BIND commands to set up your BIND environment as you did with previous releases of this product. The term UCX BIND in Table 5–1 describes the previous configuration method even though this method is still valid in the current release.

Table 5–1 describes changes to the database and configuration file names.

Table 5–1 UCX BIND and BIND 8.1 Differences

Database/File Names	UCX BIND	BIND 8.1
Configuration information	UCX$CONFIGURATION.DAT	TCPIP$BIND.CONF
Local loopback files	NAMED.LOCAL	LOCALHOST.DB, 127_0_0.DB
Forward lookup file	*domain_name*.DB	*domain_name*.DB
Reverse lookup file	*address*.DB	*address*.DB
Cache file	NAMED.CA	ROOT.HINT

Important

You must be consistent when making changes to your BIND environment. If you make changes by editing the configuration file, you should continue to make changes in that manner.

If you revert to the UCX BIND configuration method (SET CONFIGURATION BIND and CONVERT/CONFIGURATION BIND commands), any changes you made to the configuration file (TCPIP$BIND.CONF) are lost.

If you continue to use the SET CONFIGURATION BIND commands, you must always enter the CONVERT/CONFIGURATION BIND command in order for your changes to take effect.

5.3 Configuring the BIND Server (BIND 8.1)

This section describes how to configure the BIND name server on your local host.

BIND 8.1 stores configuration information in a text file called TCPIP$BIND.CONF. The TCP/IP Services product provides a template for this file located in the SYS$SPECIFIC:[TCPIP$BIND] directory. Edit this template to reflect your site-specific configuration requirements before running BIND.

A BIND 8.1 configuration file consists of statements and comments. Statements end with a semicolon. Many statements contain a block of substatements that also end with a semicolon. Table 5–2 describes the valid configuration statements. For detailed descriptions of these statements, see the BIND 8 documentation at

`www.isc.org/`

Table 5–2 BIND Name Server Configuration Statements

Statement	Description
`acl` *name*	Defines an address match list used for access control and other uses. The following ACLs are built in: `Any` Allows all hosts. `None` Denies all hosts. `Localhost` Allows the IP addresses of all interfaces on the system. `localnets` Allows any host on a network for which the system has an interface. See Section 5.3.5 for more information about these options.

`include` *path_name*	Inserts a file. Use this statement to break the configuration file into manageable sections. The following lines, for example, could be placed at the top of a BIND configuration file so that it includes any ACL or key information.
	`include "SYS$SPECIFIC:[TCPIP$BIND]KEYS.BIND";`
	`include "SYS$SPECIFIC:[TCPIP$BIND]ACLS.BIND";`

(continued on next page)

Table 5–2 (Cont.) BIND Name Server Configuration Statements

Statement	Description
`logging`	Configures logging options for the name server. Options include output methods, format options, and severity levels that you associate with a name that can then be used with the category phrase to select how various classes of messages are logged. Use one logging statement to define as many channels and categories as you want. See Section 5.3.1 for more information about these options.
`options`	Sets up global server configuration options and sets defaults for other statements. This statement is used only once in a configuration file. Options statements include path names, Boolean options, forwarding information, name checking, access control, interfaces, query addresses, zone transfers, resource limits, periodic task intervals, and topology. See Section 5.3.2 for more information about these options.
`server` *ip_address*	Defines the characteristics associated with a remote name server. The server supports the following zone transfer methods: `one-answer` Uses one DNS message per resource record transferred. `many-answers` Packs as many resource records as possible into a message. This method is more efficient but is only understood by BIND 8.1.2. You can specify which method to use for a server with the `transfer-format` option. If `transfer-format` is not specified, the `transfer-format` specified by the options statement is used.
`zone` *domain_name*	Defines a zone. Valid types include master, slave, stub, and hint.

The following sample is a configuration file for a master server:

```
options {
        directory "SYS$SPECIFIC:[TCPIP$BIND]";
};
zone "FRED.PARROT.BIRD.COM" in {
        type master;
        file "FRED_PARROT_BIRD_COM.DB";
};

zone "0.0.127.IN-ADDR.ARPA" in {
        type master;
        file "127_0_0.DB";
};
zone "LOCALHOST" in {
        type master;
        file "LOCALHOST.DB";
};
zone "208.20.16.IN-ADDR.ARPA" in {
        type master;
        file "208_20_16_IN-ADDR_ARPA.DB";
};
zone "." in {
        type hint;
        file "ROOT.HINT";
};
```

The following comment styles are valid in a BIND configuration file. Comments can appear anywhere in the file.

- C-style comments that start with /* and end with */

- C++ style comments that start with // and continue to the end of the physical line
- Shell or Perl-style comments that start with # and continue to the end of the physical line

Important

In a zone file, comments start with a semicolon (;). Do not use the semicolon as a comment character in your configuration file. The semicolon indicates the end of a configuration statement, so whatever follows is interpreted as the start of the next statement.

5.3.1 BIND Configuration Logging Statement

The logging statement configures a wide variety of logging options for the name server. Its channel phrase associates output methods, format options, and severity levels with a name that can then be used with the category phrase to select how various classes of messages are logged. The logging statement has the following syntax:

```
logging {
   [ channel channel_name {
     ( file path_name
       [ versions ( number | unlimited ) ]
       [ size size_spec ]
       | syslog ( kern | user | mail | daemon | auth | syslog | lpr |
                  news | uucp | cron | authpriv | ftp |
                local0 | local1 | local2 | local3 |
                local4 | local5 | local6 | local7 )
       | null;

   [ severity ( critical | error | warning | notice |
                  info  | debug [ level ] | dynamic ); ]
   [ print-category yes_or_no; ]
   [ print-severity yes_or_no; ]
   [ print-time yes_or_no; ]
}; ]

   [ category category_name {
     channel_name; [ channel_name; ... ]
}; ]
  ...
};
```

Only one logging statement is used to define as many channels and categories as you want. If there are multiple logging statements in a configuration file, the first one that is defined determines the logging, and warnings are issued for the others. If there is no logging statement, the logging configuration is:

```
logging {
    category default { default_syslog; default_debug; };
    category panic { default_syslog; default_stderr; };
    category packet { default_debug; };
    category eventlib { default_debug; };
};
```

All logged statements channeled to `syslog` facilities are directed to the TCPIP$BIND_RUN.LOG file.

5.3.1.1 Channel Phrase

All log output goes to one or more channels. You can create as many channels as you want.

Every channel definition must include a clause that says whether messages selected for the channel go to a file or to a particular `syslog` facility, or are discarded. Optionally, it can also limit the message severity level that is accepted by the channel (default is `info`), and whether to include a name server-generated timestamp, the category name, and severity level (default is not to include any).

The word `null` as the destination option for the channel causes all messages sent to it to be discarded; other options for the channel are meaningless.

There is a `severity` clause that allows you to specify the level of diagnostic messages to be logged.

The server can supply extensive debugging information when it is in debugging mode. If the server's global debugging level is greater than zero, then debugging mode is active. The global debugging level is set by one of the following:

- Starting the server with the "-d" flag, followed by a positive integer
- Sending the server the SIGUSR1 signal (for example, by entering SYS$SYSTEM:TCPIP$BIND_SERVER_CONTROL.EXE TRACE)

The global debugging level can be set to zero, and the debugging mode turned off, by sending the server the SIGUSR2 signal (by entering SYS$SYSTEM:TCPIP$BIND_SERVER_CONTROL.EXE NOTRACE). All debugging messages in the server have a debugging level; the higher debugging levels provide more detailed output. Channels that specify a particular debugging severity will get debugging output of level 3 or less any time the server is in debugging mode, regardless of the global debugging level. Channels with dynamic severity use the server's global level to determine what messages to display, as shown in the following example:

```
channel specific_debug_level {
    file "foo";
    severity debug 3;
};
```

If `print-time` is turned on, the date and time are logged. `print-time` can be specified for a `syslog` channel, but that is usually pointless since `syslog` also prints the date and time. If `print-category` is requested, then the category of the message is logged as well. Finally, if `print-severity` is on, then the severity level of the message is logged. The `print-` options can be used in any combination and are always displayed in the following order: time, category, severity. In the following example, all three `print-` options are on:

```
28-Apr-1997 15:05:32.863 default: notice: Ready to answer queries.
```

There are four predefined channels that are used for the BIND server's default logging, as shown in the following example. Section 5.3.1.2 describes how these channels are used.

```
channel default_syslog {
    syslog daemon;      # send to syslog's daemon facility
    severity info;      # only send priority info and higher
};
channel default_debug {
    file "TCPIP$BIND_RUN.LOG";   # write to TCPIP$BIND_RUN.LOG in the
                                 # working directory
    severity dynamic;            # log at the server's current debug level
};
channel default_stderr {# writes to stderr
    file "stderr";      # this is illustrative only; there's currently
                        # no way of specifying an internal file
                        # descriptor in the configuration language.
    severity info;      # only send priority info and higher
};

channel null {
    null;               # toss anything sent to this channel
};
```

Once a channel is defined, it cannot be redefined. Thus you cannot alter the built-in channels directly, but you can modify the default logging by pointing categories at channels you have defined.

5.3.1.2 Category Phrase

There are many categories, so you can send the logs you want to see anywhere, without seeing logs you do not want. If you do not specify a list of channels for a category, then log messages in that category are sent to the default category instead. If you do not specify a default category, the following "default default" is used:

```
category default { default_syslog; default_debug; };
```

For example, if you want to log security events to a file but you also want to keep the default logging behavior, specify the following:

```
channel my_security_channel {
    file "my_security_file";
    severity info;
};
category security { my_security_channel;
    default_syslog;
    default_debug;
};
```

To discard all messages in a category, specify the null channel:

```
category lame-servers { null; };
category cname { null; };
```

The following categories are available:

`default` The catch-all. Many things are not categorized, and they end up here. Also, if you do not specify any channels for a category, the default category is used instead. If you do not define the default category, the following definition is used:

```
category default { default_syslog; default_debug; };
```

`config` High-level configuration file processing.

`parser` Low-level configuration file processing.

`queries` A short log message is generated for every query the server receives.

lame-servers	Messages like "Lame server on ..." .
statistics	Statistics.
panic	If the server has to shut itself down because of an internal problem, it logs the problem in this category as well as in the problem's native category. If you do not define the panic category, the following definition is used: `category panic { default_syslog; default_stderr; };`
update	Dynamic updates.
ncache	Negative caching.
xfer-in	Zone transfers the server is receiving.
xfer-out	Zone transfers the server is sending.
db	All database operations.
eventlib	Debugging information from the event system. Only one channel can be specified for this category, and it must be a file channel. If you do not define the eventlib category, the following definition is used: `category eventlib { default_debug; };`
packet	Dumps of packets received and sent. Only one channel can be specified for this category, and it must be a file channel. If you do not define the packet category, the following definition is used: `category packet { default_debug; };`
notify	The NOTIFY protocol.
cname	Messages like "... points to a CNAME".
security	Approved and unapproved requests.
os	Operating system problems.
insist	Internal consistency check failures.
maintenance	Periodic maintenance events.
load	Zone loading messages.
response-checks	Messages arising from response checking, such as "Malformed response ...", "wrong ans. name ...", "unrelated additional info ...", "invalid RR type ...", and "bad referral ...".

5.3.2 BIND Configuration Options Statement

The options statement sets up global options to be used by BIND. This statement can appear only once in a configuration file; if more than one occurrence is found, the first occurrence determines the actual options used, and a warning is generated. If there is no options statement, an options block with each option set to its default is used. The options statement has the following syntax:

```
options {
[ directory path_name; ]
[ named-xfer path_name; ]
[ dump-file path_name; ]
[ pid-file path_name; ]
[ statistics-file path_name; ]
[ auth-nxdomain yes_or_no; ]
[ fake-iquery yes_or_no; ]
[ fetch-glue yes_or_no; ]
[ multiple-cnames yes_or_no; ]
[ notify yes_or_no; ]
[ recursion yes_or_no; ]
[ forward ( only | first ); ]
[ forwarders { [ in_addr ; [ in_addr ; ... ] ] }; ]
[ check-names ( master | slave | response ) ( warn | fail | ignore); ]
[ allow-query { address_match_list }; ]
[ allow-transfer { address_match_list }; ]
[ listen-on [ port ip_port ] { address_match_list }; ]
[ query-source [ address ( ip_addr | * ) ] [ port ( ip_port | * ) ] ; ]
[ max-transfer-time-in number; ]
[ transfer-format ( one-answer | many-answers ); ]
[ transfers-in  number; ]
[ transfers-out number; ]
[ transfers-per-ns number; ]
[ coresize size_spec ; ]
[ datasize size_spec ; ]
[ files size_spec ; ]
[ stacksize size_spec ; ]
[ cleaning-interval number; ]
[ interface-interval number; ]
[ statistics-interval number; ]
[ topology { address_match_list }; ]

};
```

5.3.2.1 Path Names

Table 5–3 lists the path name options.

Table 5–3 Path Name Options

Option	Description
`directory`	The working directory of the server specified as an absolute path. Any nonabsolute path names in the configuration file are relative to this directory. The default location for most server output files (such as, TCPIP$BIND_RUN.LOG) is this directory. If you do not specify a directory, the working directory defaults to SYS$SPECIFIC:[TCPIP$BIND]. If you are configuring a BIND failover environment, the working directory is defined by the logical TCPIP$BIND_COMMON.
`dump-file`	The path name of the file the server dumps the database to when it receives a SIGINT signal (TCPIP$BIND_SERVER_CONTROL.EXE dumpdb). If not specified, the default is TCPIP$BIND_SERVER_ZONES_DUMP.DB.
`memstatistics-file`	The path name of the file the server writes memory usage statistics to on exit, if `deallocate-on-exit` is yes. If not specified, the default is TCPIP$BIND_SERVER_MEMSTATISTICS.LOG.

(continued on next page)

Table 5–3 (Cont.) Path Name Options

Option	Description
pid-file	The path name of the file in which the server writes its process ID. If not specified, the default is TCPIP$BIND_SERVER.PID. The pid-file is used by programs like TCPIP$BIND_SERVER_CONTROL.EXE that want to send signals to the running name server.
statistics-file	The path name of the file the server appends statistics to when it receives a SIGILL signal (TCPIP$BIND_SERVER_CONTROL.EXE stats). If not specified, the default is TCPIP$BIND_SERVER_STATISTICS.LOG.

Example 5–1 shows how to specify path name options with the options statement.

Example 5–1 Path Name Options

```
options {
 directory   "SYS$SPECIFIC:[TCPIP$BIND]";
 dump-file  "TCPIP$BIND_SERVER_ZONES_DUMP.DB";
 memstatistics-file  "TCPIP$BIND_SERVER_MEMSTATISTICS.LOG";
 pid-file  "TCPIP$BIND_SERVER.PID";
 statistics-file  "TCPIP$BIND_SERVER_STATISTICS.LOG"
};
```

5.3.2.2 Boolean Options

Table 5–4 lists the Boolean options.

Table 5–4 Boolean Options

Option	Description
auth-nxdomain	If yes, then the AA bit is always set on NXDOMAIN responses, even if the server is not actually authoritative. The default is yes. Do not turn off auth-nxdomain unless you are knowledgeable about the option, as some older versions of software may respond unpredictably.
deallocate-on-exit	If yes, then when the server exits, it deallocates every object it allocated, and then writes a memory-usage report to the memstatistics-file. The default is no, because it is faster to let the operating system clean up. The deallocate-on-exit option is useful for detecting memory leaks.
fake-iquery	If yes, the server simulates the obsolete DNS query type IQUERY. The default is no.
fetch-glue	If yes (the default), the server fetches "glue" resource records it does not have when constructing the additional data section of a response. no-fetch-glue can be used with no-recursion to prevent the server's cache from growing or becoming corrupted (at the cost of requiring more work from the client).
host-statistics	If yes, then statistics are kept for every host that the name server interacts with. The default is no. Note that turning on host-statistics can consume huge amounts of memory.

(continued on next page)

Table 5–4 (Cont.) Boolean Options

Option	Description
multiple-cnames	If yes, then multiple CNAME resource records allow for a domain name. The default is no. Allowing multiple CNAME records is against standards and is not recommended. Multiple CNAME support is available because previous versions of BIND allowed multiple CNAME records, and these records have been used for load balancing by a number of sites.
notify	If yes (the default), DNS NOTIFY messages are sent when a zone for which the server is authoritative changes. The use of NOTIFY speeds convergence between the master and its slaves. Slave servers that receive a NOTIFY message and understand it contact the master server for the zone to see if they need to do a zone transfer; if they do, the servers initiate the zone transfer immediately. The notify option can also be specified in the zone statement, in which case it overrides the options notify statement.
recursion	If yes, and a DNS query requests recursion, then the server attempts to do all the work required to answer the query. If recursion is not on, the server returns a referral to the client if it doesn't know the answer. The default is yes. See also fetch-glue above.

Example 5–2 shows how to specify Boolean options in an options statement.

Example 5–2 Boolean Options

```
options {
 auth-nxdomain  yes;
 deallocate-on-exit no;
 fake-iquery  no;
 fetch-glue  yes;
 host-statistics  no;
 multiple-cnames  no;
 notify   yes;
 recursion  yes;
};
```

5.3.2.3 Forwarding

You can use the forwarding facility to create a large sitewide cache on a few servers, reducing traffic over links to external name servers. You can also use the forwarding facility to allow queries by servers that do not have direct access to the Internet, but wish to look up exterior names anyway. Forwarding occurs on only those queries for which the server is not authoritative and does not have the answer in its cache. Table 5–5 lists the forwarding options.

Table 5–5 Forwarding Options

Option	Description
forward	This option is meaningful only if the forwarders list is not empty.
	first — Causes the server to query the forwarders first. If that does not answer the question, the server looks for the answer itself. A ROOT.HINT file must be present. This is the default.
	only — The server queries only the forwarders. A ROOT.HINT file is not necessary.
forwarders	Specifies the IP addresses to be used for forwarding. The default is the empty list (no forwarding).

Example 5–3 shows how to specify an options statement to invoke forwarding.

Example 5–3 Forwarding Options

```
options {
 forwarders {
   1.2.3.4;
   5.6.7.8;
 };
 forward first;
};
```

5.3.2.4 Name Checking

The server can check domain names based on their expected client contexts. For example, a domain name used as a host name can be checked for compliance with the RFCs defining valid host names. Table 5–6 describes the three name-checking methods.

Table 5–6 Name Checking Options

Option	Description
ignore	No checking is done.
warn	Names are checked against their expected client contexts. Invalid names are logged, but processing continues normally.
fail	Names are checked against their expected client contexts. Invalid names are logged, and the offending data is rejected.

The server can check names in three areas: master zone files, slave zone files to queries the server has initiated. If check-names response fail has been specified, and answering the client's question would require sending an invalid name to the client, the server sends a REFUSED response code to the client. The defaults are:

- check-names master fail
- check-names slave warn
- check-names response ignore

The check-names option can also be specified in the zone statement, in which case it overrides the options check-names statement. When used in a zone statement, the area is not specified (because it can be deduced from the zone type).

Example 5–4 shows how to specify an options statement for name checking. The statement specifies that nonconforming names coming from a slave are ignored.

Example 5–4 Name Checking Options

```
options  {
          check-names slave ignore;
};
```

5.3.2.5 Access Control

Access to the server can be restricted based on the IP address of the requesting system. Table 5–7 describes the access control options.

Table 5–7 Access Control Options

Option	Description
allow-query	Specifies which hosts are allowed to ask ordinary questions. The allow-query option can also be specified in the zone statement, in which case it overrides the option allow-query statement. If not specified, the default is to allow queries from all hosts.
allow-transfer	Specifies which hosts are allowed to receive zone transfers from the server. The allow-transfer option can also be specified in the zone statement, in which case it overrides the option allow-transfer statement. If not specified, the default is to allow transfers from all hosts.

Example 5–5 shows how to specify an options statement to control access to the server.

Example 5–5 Access Control Options

```
options {
 allow-query { any; } ;
 allow-transfer {
       1.2.3/24;
      5.6.7.8;
 };
};
```

5.3.2.6 Interfaces

The interfaces and ports from which the server answers queries can be specified using the listen-on option. The listen-on option takes an optional port and an address_match_list. The server listens on all interfaces allowed by the address match list. All queries must be directed to the interface and port number specified on the listen-on statement. If a port is not specified, the server uses port 53.

Multiple listen-on statements are allowed. For example,

```
listen-on { 5.6.7.8; };
listen-on port 1234 { !1.2.3.4; 1.2/16; };
```

If a listen-on option is not specified, the server listens on port 53 on all interfaces.

5.3.2.7 Query Address

If the server does not know the answer to a question, it queries other name servers. The query-source option specifies the source address and source port used for such queries. If the address is an asterisk (*) or is omitted, the server uses a wildcard IP address (INADDR_ANY). If the port is an asterisk (*) or is omitted, the server uses a random unprivileged port. The default query-source statement is as follows:

```
query-source address * port *;
```

The query-source option currently applies only to UDP queries; TCP queries always use a wildcard IP address and a random unprivileged port.

5.3.2.8 Zone Transfers

Table 5–8 describes the zone transfer options.

Table 5–8 Zone Transfer Options

Option	Description
max-transfer-time-in	Inbound zone transfers (BIND_SERVER_XFER processes) running longer than this many minutes are terminated. The default is 120 minutes (2 hours).
transfer-format	The server supports two zone transfer methods. The one-answer method uses one DNS message per resource record transferred. The many-answers method packs as many resource records as possible into a message. The many-answers method is more efficient, but it is understood only by BIND 8.1 and patched versions of BIND 4.9.5. The default is one-answer. The transfer-format option can be overridden on a per-server basis by using the server statement.
transfers-in	The maximum number of inbound zone transfers that can be running concurrently. The default value is 10. Increasing transfers-in can speed the convergence of slave zones, but can also increase the load on the local system.
transfers-per-ns	The maximum number of inbound zone transfers (BIND_SERVER_XFER processes) that can be concurrently transferring from a given remote name server. The default value is 2. Increasing transfers-per-ns can speed the convergence of slave zones, but it can also increase the load on the remote name server. transfers-per-ns can be overridden on a per-server basis by using the transfers phrase of the server statement.

Example 5–6 shows how to specify an `options` statement to control zone transfers.

Example 5–6 Zone Transfer Options

```
options {
 max-transfer-time-in 120;
 transfer-format  one-answer;
 transfers-in  10;
 transfers-per-ns 2;
};
```

5.3.2.9 Periodic Task Intervals

Table 5–9 describes the periodic task options.

Table 5–9 Periodic Task Options

Option	Description
`cleaning-interval`	The server removes expired resource records from the cache every `cleaning-interval` minutes. The default is 60 minutes. If set to 0, no periodic cleaning occurs.
`interface-interval`	The server scans the network interface list every `interface-interval` minutes to see if interfaces have been added or deleted. The default is 60 minutes. If set to 0, only interface scanning occurs when the configuration file is loaded. After the scan, listeners are started on any new interfaces (provided they are allowed by the listen-on configuration). Listeners are deleted for any interface that has gone away.
`statistics-interval`	Name-server statistics are logged every `statistics-interval` minutes. The default is 60. If set to 0, no statistics are logged.

5.3.2.10 Topology

If other settings are the same, when the server chooses a name server to query from a list of name servers, it chooses the one that is topologically closest to itself. The topology statement takes an `address_match_list` and interprets it in a special way. Each top-level list element is assigned a distance. Non-negated elements get a distance based on their position in the list; the closer the match is to the start of the list, the shorter the distance is between the match and the server. A negated match is assigned the maximum distance from the server. If there is no match, the address gets a distance that is further than any non-negated list element and closer than any negated element.

In the following example, the server prefers servers on network 10 the most, followed by hosts on network 1.2.0.0 (netmask 255.255.0.0) and network 3, with the exception of hosts on network 1.2.3 (netmask 255.255.255.0), which is preferred least of all.

```
topology {
    10/8;
    !1.2.3/24;
    { 1.2/16; 3/8; };
};
```

The default topology is as follows:

```
topology { localhost; localnets; };
```

5.3.3 BIND Configuration Server Statement

Zone transfers can put a heavy load on network traffic and on a BIND server. If you have a large network with many BIND servers, keeping each server up-to-date can put a strain on the master server and its memory requirements. You can use the server statement to control the number of zone transfers that can occur and the duration of the zone transfer.

The server statement defines characteristics to be associated with a remote name server. The statement has the following syntax:

```
server ip_address {
          [ bogus yes_or_no; ]
          [ transfers number; ]
          [ transfer-format ( one-answer | many-answers
) ; ]
          [ keys { key_id [key_id ... ] }; ]
};
```

5.3.3.1 Limiting the Number of Transfers

You can limit the number of zones your name server requests from a single remote name server by including a transfers substatement on the server statement. The default limit is two active zone transfers per name server. When your name server completely receives one of the two zone transfers, if another request is needed, the request will be sent after waiting a period of time.

Example 5–7 shows how to code the server statement to limit the number of transfer requests to name server 16.168.1.2 to three at a time.

Example 5–7 Server Statement

```
server 16.168.1.2 {
        transfers 3
};
```

5.3.3.2 Efficient Zone Transfers

The BIND 8 transfer-format subcommand specifies how your name server transfers zone data to its slaves. The server can transfer one resource record in each DNS message (one-answer) or it can transfer as many records as possible into a single DNS message (many-answers). Using the many-answers method takes less bandwidth to transfer the same amount of data as the one-answer method and also uses less CPU time to decouple the DNS messages.

Example 5–8 shows how to code the server statement to send more than one record in the same DNS message to name server 16.168.1.2.

Example 5–8 Server Statement

```
server 16.168.1.2 {
        transfer-format many-answers;
};
```

5.3.4 BIND Configuration Zone Statement

The zone statement defines zones maintained by the name server. The statement has the following syntax:

```
zone domain_name [ ( in | hs | hesiod | chaos ) ] {
                    type master;
                    file path_name;
                    [ check-names ( warn | fail | ignore ); ]
                    [ allow-update { address_match_list }; ]
                    [ allow-query { address_match_list }; ]
                    [ allow-transfer { address_match_list }; ]
                    [ notify yes_or_no; ]
                    [ forward ( only | first ); ]
                    [ also-notify { ip_addr; [ ip_addr; ... ] };
                  };
zone domain_name [ ( in | hs | hesiod | chaos ) ] {
                    type ( slave | stub );
                    [ file path_name; ]
                    masters [ port ip_port ] { ip_addr; [ ip_addr; ... ] };
                    [ check-names ( warn | fail | ignore ); ]
                    [ allow-update { address_match_list }; ]
                    [ allow-query { address_match_list }; ]
                    [ allow-transfer { address_match_list }; ]
                    [ max-transfer-time-in number; ]
                    [ notify yes_or_no; ]
                    [ also-notify { ip_addr; [ ip_addr; ... ] };
                  };

zone "." [ ( in | hs | hesiod | chaos ) ] {
                    type hint;
                    file path_name;
                    [ check-names ( warn | fail | ignore ); ]
                  };
```

5.3.5 Address Match Lists and ACLs

BIND 8.1 uses address match lists for security. Address match lists are lists of elements that can include the following:

- An IP address (in dotted-decimal notation)
- An IP prefix (in the /-notation)
- The name of an address match list previously defined with the `acl` statement
- An IP address match list

The ACLs `any`, `none`, `localhost`, and `localnets` are predefined. Elements can be negated with a leading !.

When a given IP address or prefix is compared to an address match list, the list is traversed in order, and the first match (regardless of negation) is used. The interpretation of a match depends on whether the list is being used for access control or as a topology.

When used as an access control list, a non-negated match allows access, and a negated match denies access. If there is no match, access is denied. The clauses `allow-query`, `allow-transfer`, and `allow-update` all use address match lists like this. Similarly, the `listen-on` clause can use negation to define local addresses that should not be used to accept name server connections.

When used with the topology clause, a non-negated match returns a distance based on its position on the list. (The closer the match is to the start of the list, the shorter the distance is between the match and the server.) A negated match is assigned the maximum distance from the server. If there is no match, the address gets a distance that is further than any non-negated list element and closer than any negated element.

Because of the first-match aspect of the algorithm, an element that defines a subset of another element in the list should come before the broader element, regardless of whether either one is negated. For example, in 1.2.3/24; ! 1.2.3.13; the 1.2.3.13 element is useless because the algorithm matches any lookup for 1.2.3.13 to the 1.2.3/24 element. Using ! 1.2.3.13; 1.2.3/24 fixes that problem by having 1.2.3.13 blocked by the negation, but ignores all the other 1.2.3.* hosts.

5.3.6 Dynamic Updates

BIND 8.1 includes support for dynamic updates as specified in RFC 2136 (excluding support for the security mechanism described by RFC 2137). Any update requests received from hosts that are on the server's allowed list are honored. Dynamic updates allow the addition or deletion of resource records (RR) and RR sets from a specified zone.

By default, BIND 8.1 servers reject all dynamic update requests. This is a security mechanism that gives the zone administrator the ability to decide which hosts can submit dynamic updates. You specify the hosts from which a server will process requests by using the allow-update substatement. The allow-update substatement is applicable to a zone. You cannot specify this substatement as part of an options statement.

The syntax of the allow-update substatement is as follows:

```
allow-update { address_match_list } ;
```

The following example shows the use of the allow_update substatement:

```
zone "FRED.PARROT.BIRD.COM" in {
        type master;
        file "FRED_PARROT_BIRD_COM.DB";
        allow-update {
                     99.1.2.3;
                     99.4.5.6;
        }
}
```

IP addresses, IP prefixes, ACLs, and IP address match lists are all valid elements for the allow-update substatement.

When dynamic updates are sent to and accepted by a name server, the name server does the following:

- Adds the updates to (or deletes the updates from) the memory cache copy of the zone's resource records.
- Saves the updates to a transaction log file. The default name for this file is *domain_name*.DB_LOG.

- Scans the transaction log file once per hour and updates the *domain_name*.DB file with any transactions it finds by writing a new version of the *domain_name*.DB file to disk. This action does not preserve the formatting or comments that existed in the original *domain_name*.DB file. (See Section 5.3.6.1 for solutions for preserving the formatting or comments in the original *domain_name*.DB file.)
- Renames the current transaction log file to *domain_name*.DB_LOG_BCK and then creates a new *domain_name*.DB_LOG file.

5.3.6.1 Preserving the Zone File

Typically, a system administrator adds comments to the *domain_name*.DB file to provide history and helpful information pertinent to the data in the file, and formats the file for easy reading. With DNS dynamic updates enabled, all comments, formatting, and ordering will be lost.

TCP/IP Services provides two methods to prevent this:

- You can use the DCL command DEFINE to define the logical TCPIP$BIND_DONT_MERGE_DYNAMIC_UPDATES. The presence of the logical turns off the merge of dynamic updates, and the server does not create new versions of the *domain_name*.DB.
- TCP/IP Services BIND server preserves the original *domain_name*.DB file that is read when the BIND server starts up. The server never deletes or purges the original database file.

The BIND server is able to detect a situation in which dynamic updates might be lost. When this happens, the server creates a new version of the *domain_name*.DB_LOG_BCK file containing the dynamic updates that would have been lost. The system administrator must review the transactions in the file and determine whether the updates are still valid and if so, manually apply the updates to the *domain_name*.DB file.

There is always at least one version of the *domain_name*.DB_LOG_BCK file when dynamic updates are enabled. Each time the BIND server detects lost updates, the server creates a new version of the *domain_name*.DB_LOG_BCK file. The existence of more than one of these files is a signal to the system administrator that manual merges may be necessary.

The server does not automatically purge the *domain_name*.DB_LOG_BCK files, but the system administrator can delete them after examining and applying their contents.

5.3.6.2 Manually Creating Updates

You can manually create updates to the domain database file using the command line utility NSUPDATE if the name server for the domain is configured to accept dynamic updates.

The format of the NSUPDATE command is:

```
NSUPDATE [ -d ] [ -v ] [ file-name ]
```

In this format:

-d	Specifies debug mode.
-v	Specifies that NSUPDATE uses the TCP protocol instead of the UDP protocol.
file-name	Specifies a file name containing update requests and entries.

Table 5–10 shows the valid update commands for NSUPDATE.

Table 5–10 NSUPDATE Commands

Command	Description
prereq yxrset *domain_name type* [*rdata*]	Makes the presence of an RR set of *type* owned by *domain_name* a prerequisite to performing the update.
prereq nxrrset	Makes the nonexistence of an RR set of *type* owned by *domain_name* a prerequisite to performing the update specified in successive update commands.
prereq yxdomain *domain_name*	Makes the existence of the specified *domain_name* a prerequisite to performing the update.
prereq nxdomain	Makes the nonexistence of the specified domain name a prerequisite to performing the update
update delete *domain_name* [*type*] [*rdata*]	Deletes the specified domain name, or, if *type* is also specified, deletes the specified RR set, or, if *rdata* is also specified, deletes the record matching *domain_name*, *type*, and *rdata*.
update add *domain_name ttl* [*class*] *type rdata*	Adds the specified record to the zone. Note that the TTL, in addition to the type and resource-record-specific data, must be included but *class* is optional and defaults to IN.

NSUPDATE has two modes: interactive and noninteractive. In noninteractive mode, you supply the updates in a file. Data in the file must be in the following format:

class section name ttl type rdata

In this format:

class	Is any one of the following opcodes: update, zone, prereq.
section	Is any one of the following opcodes: add, delete, nxdomain, yxdomain, nxrrset, yxrrset.
name	Is the name of the entry being added.
ttl	Is time to live (in seconds) for this entry. After this time period, the name server will no longer serve the entry.
type	Specifies the RR type (for example, A, CNAME, NS, MX, TXT).
rdata	Specifies the data appropriate for the RR type being updated.

The following example shows how to use NSUPDATE in the noninteractive mode.

```
$ TYPE NSUPD.TXT
update delete www.nads.zn.
update add www.nads.zn. 60 CNAME ivy18.nads.zn
$ NSUPDATE NSUPD.TXT
```

In interactive mode, you supply data in the format shown for noninteractive mode in response to each NSUPDATE prompt.

The following example shows how to use the NSUPDATE utility in interactive mode. The Resolver debug mode is enabled.

```
$ NSUPDATE
> UPDATE ADD WWW.NADS.ZN 60 IN CNAME IVY18.NADS.ZN
>
res_mkupdate: packet size = 49
;; res_send()
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id: 53349
;; flags:; ZONE: 1, PREREQUISITE: 0, UPDATE: 1, ADDITIONAL: 0
;;      nads.zn, type = SOA, class = IN
www.nads.zn.            1M IN CNAME     ivy18.nads.zn.
;; Querying server (# 1) address = 192.168.1.1
;; got answer:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id: 53349
;; flags: qr ra; ZONE: 0, PREREQUISITE: 0, UPDATE: 0, ADDITIONAL: 0
```

Note

When entering data in interactive mode, pressing Return (or Enter) with no data assumes the end of the input. NSUPDATE then processes all update entries in one operation.

5.3.7 Configuring Cluster Failover and Redundancy

In the same OpenVMS Cluster, multiple BIND master servers can share a common database, thereby providing redundancy and a failover mechanism when one of the servers becomes unavailable.

To configure a DNS cluster failover and redundancy environment, perform the following steps on each node participating in the cluster.

1. Run the TCPIP$CONFIG command procedure, and from the Servers menu enable the BIND service.
2. Edit the BIND configuration file, SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND.CONF.
   - Configure the node as a master server.
   - Add or edit the `options` statement. The `directory` substatement should be as follows:

     ```
     options {
             directory "TCPIP$BIND_COMMON";
     };
     ```

     TCPIP$BIND_COMMON is a logical name defined in the TCPIP$BIND_COMMON_STARTUP.COM command procedure as a search list. The search list consists of the SYS$SPECIFIC:[TCPIP$BIND] directory and the common directory. You will be prompted by the setup command procedure in the next step to specify the device on which the common directory is to reside. If you do not specify a device, the default device and directory is *common_device*:[TCPIP$BIND_COMMON], where *common_device* is automatically generated in the following manner:

     - If the SYSUAF logical is defined, the common disk is determined from its definition.
     - If the SYSUAF logical is not defined, the system uses SYS$SYSDEVICE as the default device.

3. Run the SYS$COMMON:[SYSMGR]TCPIP$BIND_CLUSTER_SETUP.COM command procedure.

   This procedure creates two other command procedures that manage the startup and shutdown processes of the BIND component in a cluster environment:

   - SYS$COMMON:[SYSMGR]TCPIP$BIND_COMMON_STARTUP.COM
   - SYS$COMMON:[SYSMGR]TCPIP$BIND_COMMON_SHUTDOWN.COM

   These files define the BIND system logicals and accounting information. To remove the failover setup from your system, delete these two files.

4. Place any database files to be shared in the common directory.

   Note

   Be careful to remove from SYS$SPECIFIC:[BIND] any databases that are to be shared. Using the search list logical, BIND will find any SYS$SPECIFIC:[BIND] databases first and use those. This may not be the result you want.

5. Start up BIND by entering the following command:

   ```
   $ @SYS$COMMON:[SYSMGR]TCPIP$BIND_STARTUP.COM
   ```

   Caution

   The use of dynamic updates in conjunction with a master BIND server participating in cluster failover and redundancy is not supported and may cause serious problems.

5.3.7.1 Changing the BIND Database

If multiple master BIND servers are running in a cluster, and a change is made to the common BIND database, the database must be reloaded on each node running the master BIND server. To reload the BIND database on every node in the cluster where the master BIND server is running, enter the following command:

```
TCPIP> SET NAME_SERVICE /INITIALIZE /CLUSTER=dev:[directory]
```

The /CLUSTER qualifier takes the directory specification of the common BIND directory as a value. If you omit the device and directory, it defaults to:

common_device:[TCPIP$BIND_COMMON]

In this case, *common_device* is automatically generated in the following manner:

- If the SYSUAF logical is defined, the the common disk is determined from its definition.
- If SYSUAF logical is not defined, the system uses SYS$SYSDEVICE as the default device.

5.4 Populating the BIND Server Databases

To populate the BIND server database files, use one of the following methods:

- Convert an existing host database with the CONVERT/UNIX BIND command.
- Manually edit the ZONE.DB files.

5.4.1 Using Existing Databases

To populate the BIND server database by copying information from the hosts database and other database files, enter the CONVERT/UNIX BIND command. This command:

- Creates a BIND server database (if needed).
- Extracts data from the hosts database. (The BIND server uses UNIX style formatted files.)
- Extracts Mail Exchange (MX) information from the routes database.
- Populates the BIND server database with the host and MX records.
- Creates a forward translation file with the following characteristics:
  - It has address, canonical name, and MX entries.
  - If a file with the same name as the output file already exists, the serial number from that file's start-of-authority (SOA) entry increments and becomes the serial number of the new output file.
  - If no previous version of the output file exists, the serial number for the new file is 1.

  When you specify forward translation (by omitting the /DOMAIN qualifier), any host in the hosts database that is not qualified with a domain is included in the target domain. For example, if the local domain is `x.y.z.`, the CONVERT/UNIX BIND command includes: `a`, `b.x.y.z`, `c.x.y.z.z` but does not include `d.x.y.h`.
- Creates a reverse translation file if you specify /DOMAIN=(*domain.name*) and the end of *domain.name* is IN-ADDR.ARPA.

  The created reverse translation file has the following characteristics:
  - Only records applicable to the domain you specify are placed into the output file.
  - The output file has domain name pointer entries.
  - If a file with the same name as the output file already exists, the serial number from that file's SOA entry increments and becomes the serial number of the new output file.
  - If no previous version of the output file exists, the serial number for the new file is 1.
  - Selects hosts with IP addresses that match the partial IP address from *domain.name*. For example, /DOMAIN=16.99.IN-ADDR.ARPA does a reverse translation and selects hosts whose addresses begin with 99.16.

If the BIND server's directory is SYS$SPECIFIC:[TCPIP$BIND] and you have specified domain `abc.def.com`, the default output file is named SYS$SPECIFIC:[TCPIP$BIND]ABC_DEF_COM.DB.

Compaq suggests that you do not change the default directory name. If you do, the file is created in your current directory.

On the command line, specify the full OpenVMS file specification. Do not specify a version number, and do not use wildcards. The following example uses the domain ucx.ern.sea.com, creates a UCX_ERN_SEA_COM.DB file, creates a 208_20_9_IN-ADDR_ARPA.DB file, and checks the results by displaying directory listings with the new file.

```
TCPIP> CONVERT/UNIX BIND /DOMAIN=UCX.ERN.SEA.COM
TCPIP> CONVERT/UNIX BIND /DOMAIN=208.20.9.IN-ADDR.ARPA

TCPIP> SET DEFAULT SYS$SPECIFIC:[TCPIP$BIND]
$ DIRECTORY

Directory SYS$SPECIFIC:[TCPIP$BIND]

127_0_0.DB;1        208_20_9_IN-ADDR_ARPA.DB;1
LOCALHOST.DB;1
LOGIN.COM;1         ROOT.HINT;1         TCPIP$BIND.CONF;1
TCPIP$BIND_CONF.TEMPLATE;1              TCPIP$BIND_RUN.LOG;4339
TCPIP$BIND_SERVER.PID;1                 UCX_ERN_SEA_COM.DB;5
```

5.4.2 Manually Editing Zone Files

All name server zone files use the same type of records to define domain database information. Compaq recommends that you review these resource records before editing any BIND files. The standard resource records (RR) are summarized in Table 5–11.

Table 5–11 Standard Resource Record Types

Record Type	Description
SOA	Start of authority. Marks the beginning of a zone's data and defines parameters that affect the entire zone.
NS	Name server. Identifies a domain's name server.
A	Address. Maps a host name to an address.
PTR	Pointer. Maps an address to a host name.
MX	Mail Exchange. Identifies where to deliver mail for a given domain.
CNAME	Canonical name. Defines an alias host name.
HINFO	Host information. Describes a host's hardware and operating system.
WKS	Well-known service. Advertises network services.

The format of DNS records is as follows:

[*name*] [*ttl*] IN *type data*

In this format:

name — Specifies the name of the domain object referenced by a resource record. The string entered for *name* is the current domain unless it ends with a dot. If the name field is blank, the record applies to the domain object last named.

ttl — Defines the length of time, in seconds, that the information in this resource record should be kept in cache. Usually, the time-to-live field is left blank, and the default *ttl*, set for the entire zone SOA record, is used.

IN	Identifies the record as an Internet DNS resource record.
type	Identifies what kind of resource record this is. (See Table 5–11 for the record types you can specify.)
data	Information specific to this type of resource record. For example, in an A record, this is the field that contains the actual IP address.

5.4.3 Saving Backup Copies of Zone Data

The name server saves backup copies of the zone data in SYS$SPECIFIC:[TCPIP$BIND]. Do not delete these backup copies. When the master server is down and the secondary server is started, the secondary server cannot perform a zone transfer until the master server is up. However, with backup copies, the secondary server has some data (though possibly out of date) to perform its basic tasks.

5.4.4 Sample Database Files

The following sections provide sample BIND database files.

5.4.4.1 Local Loopback: Forward and Reverse Translation Files

In the LOCALHOST.DB file, the local host address is usually 127.0.0.1. The following sample LOCALHOST.DB file shows the forward translation for the local loopback interface.

```
;
; BIND data file for local loopback interface (forward
translation).
;
; Provided for Compaq TCP/IP Services for OpenVMS.
;
$ORIGIN localhost.
@               1D IN SOA       @ root (
                                42              ;Serial
                                3H              ;Refresh
                                15M             ;Retry
                                1W              ;Expiry
                                1D )            ;Minimum
;
                1D IN NS        @
                1D IN A         127.0.0.1
```

The following sample 127_0_0.DB file shows the reverse translation for the local loopback interface.

```
;
; BIND data file for local loopback interface (reverse
translation).
;
; Provided for Compaq TCP/IP Services for OpenVMS.
;
$ORIGIN 0.0.127.in-addr.arpa.
@               1D IN SOA       localhost.
root.localhost. (
                                42              ;Serial
                                3H              ;Refresh
                                15M             ;Retry
                                1W              ;Expiry
                                1D )            ;Minimum
;
                1D IN NS        localhost.
1               1D IN PTR       localhost.
```

These local host databases provide forward and inverse translation for the widely used LOCALHOST name. The LOCALHOST name is always associated with the IP address 127.0.0.1 and is used for local loopback traffic.

5.4.4.2 Hint File

This file contains root name server hints. Any name server running on a host without direct Internet connectivity should list the internal roots in its hint file.

The following sample shows a ROOT.HINT file. In earlier releases, this file was called NAMED.CA.

```
; Data file for initial cache data for root domain servers.
;
; Provided for Compaq TCP/IP Services for OpenVMS.
;
; <<>> DiG 8.1 <<>> @192.5.5.241
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;;      ., type = NS, class = IN
;
;; ANSWER SECTION:
.             6D IN NS H.ROOT-SERVERS.NET.
.             6D IN NS B.ROOT-SERVERS.NET.
.             6D IN NS C.ROOT-SERVERS.NET.
.             6D IN NS D.ROOT-SERVERS.NET.
.             6D IN NS E.ROOT-SERVERS.NET.
.             6D IN NS I.ROOT-SERVERS.NET.
.             6D IN NS F.ROOT-SERVERS.NET.
.             6D IN NS G.ROOT-SERVERS.NET.
.             6D IN NS J.ROOT-SERVERS.NET.
.             6D IN NS K.ROOT-SERVERS.NET.
.             6D IN NS L.ROOT-SERVERS.NET.
.             6D IN NS M.ROOT-SERVERS.NET.
.             6D IN NS A.ROOT-SERVERS.NET.
;
;; ADDITIONAL SECTION:
H.ROOT-SERVERS.NET.     5w6d16h IN A    128.63.2.53
B.ROOT-SERVERS.NET.     5w6d16h IN A    128.9.0.107
C.ROOT-SERVERS.NET.     5w6d16h IN A    192.33.4.12
D.ROOT-SERVERS.NET.     5w6d16h IN A    128.8.10.90
E.ROOT-SERVERS.NET.     5w6d16h IN A    192.203.230.10
I.ROOT-SERVERS.NET.     5w6d16h IN A    192.36.148.17
F.ROOT-SERVERS.NET.     5w6d16h IN A    192.5.5.241
G.ROOT-SERVERS.NET.     5w6d16h IN A    192.112.36.4
J.ROOT-SERVERS.NET.     5w6d16h IN A    198.41.0.10
K.ROOT-SERVERS.NET.     5w6d16h IN A    193.0.14.129
L.ROOT-SERVERS.NET.     5w6d16h IN A    198.32.64.12
M.ROOT-SERVERS.NET.     5w6d16h IN A    202.12.27.33
A.ROOT-SERVERS.NET.     5w6d16h IN A    198.41.0.4
;
;; Total query time: 608 msec
;; FROM: ucx.ern.sea.com to SERVER: 192.5.5.241
;; WHEN: Mon May 18 15:26:19 1998
;; MSG SIZE  sent: 17  rcvd: 436
```

This cache initialization file contains NS records that name root servers and A records that provide the addresses of root servers.

To create a ROOT.HINT file:

1. Run TCPIP$CONFIG.
2. Select the Server Components menu.
3. Select the BIND server.

This procedure creates the ROOT.HINT file and places the file in the SYS$SPECIFIC:[TCPIP$BIND] directory.

5.4.4.3 Forward Translation File

The forward translation file, *domain_name*.DB, stores host-name-to-address mapping. For example, for the domain ROBIN.BIRD.COM, the following database file is created: ROBIN_BIRD_COM.DB. The following example shows a *domain_name*.DB file:

```
$ORIGIN ucx.ern.sea.com.
@               IN      SOA     owl.ucx.ern.sea.com. pmaster.owl.ern.sea.com.
(
                                23      ; Serial
                                600     ; Refresh
                                300     ; Retry
                                172800  ; Expire
                                43200 ) ; Minimum
;
                IN      NS      owl.ucx.ern.sea.com.
                IN      NS      condor.ucx.ern.sea.com.
;
thrush          IN      A       9.20.208.53
condor          IN      A       9.20.208.10
birdy           IN      A       9.20.208.47
                IN      MX      10 birdy.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
seagull         IN      A       9.20.208.30
                IN      MX      10 seagull.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
owl             IN      A       9.20.208.72
                IN      MX      10 owl.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
peacock         IN      A       9.20.208.73
                IN      MX      10 pultdown.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
redwing         IN      A       9.20.208.79
                IN      MX      10 redwing.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
robin           IN      A       9.20.208.47
                IN      A       9.20.208.30
                IN      A       9.20.208.72
```

This file is created only for the master server. All other servers obtain this information from the master server. This file contains most of the domain information and has the following characteristics:

- Begins with an SOA record and a few NS records that define the domain and its servers.
- Maps host names to IP addresses.
- Contains A, MX, CNAME, and other records.

MX records identify the servers in a domain that are used for forwarding mail. Use MX records and preference numbers to define the order in which mail servers are used. The lower the preference number, the more desirable the server.

5.4.4.4 Reverse Translation File

The reverse translation file, *address*.DB, stores address-to-host-name mapping (reverse mapping) information. For example, for the same domain, a file with the name 208_20_9_IN-ADDR_ARPA.DB is created.

The following example shows an *address*.DB file.

```
$ORIGIN 208.20.9.in-addr.arpa.
@     IN   SOA   owl.ucx.ern.sea.com. pmaster.owl.ucx.ern.sea.com.
(
                          1       ; Serial
                          600     ; Refresh
                          300     ; Retry
                          172800  ; Expire
                          43200 ) ; Minimum
;
      IN      NS      owl.ucx.ern.sea.com.
      IN      NS      condor.ucx.ern.sea.com.
;
53              IN      PTR     thrush.ucx.ern.sea.com.
10              IN      PTR     condor.ucx.ern.sea.com.
47              IN      PTR     birdy.ucx.ern.sea.com.
30              IN      PTR     seagull.ucx.ern.sea.com.
72              IN      PTR     owl.ucx.ern.sea.com.
73              IN      PTR     peacock.ucx.ern.sea.com.
79              IN      PTR     redwing.ucx.ern.sea.com.
```

PTR records predominate in this file because they are used to translate addresses to host names.

5.5 Examining Name Server Statistics

The BIND server collects statistics that record server activity. To examine BIND statistics, use the SHOW NAME_SERVICE/STATISTICS command. This command logs statistics to the file TCPIP$BIND_SERVER_STATISTICS.LOG, located in SYS$SPECIFIC:[TCPIP$BIND].

The following sample shows a statistics log:

```
34250   time since boot (secs)
15670   time since reset (secs)
12      Unknown query types
20000   A queries
540     SOA queries
2399    MX queries
867     ANY queries
3       AXFR queries
```

```
++ Name Server Statistics ++
(Legend)
        RR      RNXD    RFwdR   RDupR   RFail
        RFErr   RErr    RAXFR   RLame   ROpts
        SSysQ   SAns    SFwdQ   SDupQ   SErr
        RQ      RIQ     RFwdQ   RDupQ   RTCP
        SFwdR   SFail   SFErr   SNaAns  SNXD
(Global)
        2 0 0 0 0  0 0 0 0 0  2 0 0 0 0  0 0 0 0 5  0 0 0 0 0
-- Name Server Statistics --
++ Memory Statistics ++
      3:           9 gets,           2 rem
      4:           7 gets,           0 rem (1 bl, 1022
      5:          16 gets,           1 rem
      6:           7 gets,           5 rem
      7:          10 gets,           5 rem
      8:          97 gets,          16 rem (1 bl, 485 ff)
     13:           6 gets,           4 rem
.
.
.
    664:           5 gets,           1 rem (1 bl, 5 ff)
    732:           2 gets,           0 rem (1 bl, 5 ff)
   1040:           1 gets,           1 rem (1 bl, 2 ff)
>= 1100:          23 gets,           9 rem
-- Memory Statistics --
--- Statistics Dump --- (907337687) Fri Jan 7  10:14:47 2000
```

The log lists information about how long the server has been running and how long since the last reset, and provides a count of the number of queries processed for each available resource record type.

In the Memory Statistics section, statistics for each size are displayed showing, in the following order:

1. The total number of times that a buffer of that size was allocated (gets).
2. The number of buffers of that size which have not yet been freed and currently remain allocated (rem).
3. The number of blocks currently allocated. Note that buffers are allocated in large groups at a time, and these groups are called blocks (bl).
4. The number of buffers within those blocks that are currently free (ff, which stands for free fragments).

5.6 Configuring BIND with SET CONFIGURATION Commands

The following sections describe how to manually set up BIND servers by using SET CONFIGURATION BIND commands.

Important

These commands create a UCX Version 4.*x* configuration. If you set up your BIND name server using these commands, you must also use the CONVERT/CONFIGURATION BIND command to convert the databases to the BIND 8.1 format. If you omit this step, your changes will not take effect.

5.6.1 Setting Up a Master Name Server

To instruct the master name server to read the appropriate database files using the information in TCPIP$CONFIGURATION.DAT, use the SET CONFIGURATION BIND command. Use the SHOW CONFIGURATION BIND command to display BIND information from the configuration database (TCPIP$CONFIGURATION.DAT).

The following commands tell the name server to read the appropriate files:

Example 5–9 Reading Database Files

```
TCPIP> SET CONFIGURATION BIND /CACHE

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:0.0.127.IN-ADDR.ARPA, FILE:NAMED.LOCAL)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:UCX.ERN.SEA.COM, FILE:UCX_ERN_SEA_COM.DB)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:208.20.9.IN-ADDR.ARPA, FILE:208_20_9_IN-ADDR_ARPA.DB)
```

To view these settings, use the SHOW CONFIGURATION BIND command.

5.6.2 Setting Up a Secondary (Slave) Name Server

Configure a secondary server to populate itself by copying the DNS database files from the master server.

To configure a secondary server, enter the following commands:

```
TCPIP> SET CONFIGURATION BIND /CACHE

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:0.0.127.IN-ADDR.ARPA, FILE:NAMED.LOCAL)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /SECONDARY=(DOMAIN:UCX.ERN.SEA.COM, -
_TCPIP>            FILE:UCX_ERN_SEA_COM.DB, -
_TCPIP>            HOST:OWL)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /SECONDARY=(DOMAIN:208.20.9.IN-ADDR.ARPA, -
_TCPIP>             FILE:208_20_9_IN-ADDR_ARPA.DB, -
_TCPIP>             HOST:OWL.UCX.ERN.SEA.COM))
```

5.6.3 Setting Up a Cache-Only Server

To configure a cache-only server, enter:

```
TCPIP> SET CONFIGURATION BIND /CACHE
```

This command points the server to the file NAMED.CA.

5.6.4 Setting Up a Forwarder Name Server

To configure a forwarder, enter the following command:

```
TCPIP> SET CONFIGURATION BIND /FORWARDERS=(HOST:host)
```

In this command, *host* specifies the forwarding server.

Note

You cannot set up a server to be both a forwarder and a caching server.

5.7 Configuring the BIND Resolver

Your host uses the BIND resolver to obtain information from a name server. When a request for name translation arrives, the resolver first searches the local host database for the host information. If not found, the resolver then queries the BIND name server for host information.

The resolver is automatically configured by TCPIP$CONFIG when you choose "Option 1 — Core Environment." To display your resolver configuration, enter the following command:

```
TCPIP> SHOW NAME_SERVICE
```

TCP/IP Services displays the following data:

```
BIND Resolver Parameters

 Local domain: ucx.ern.sea.com

 System

  State:     Started, Enabled

  Transport: UDP
  Domain:    ucx.ern.sea.com
  Retry:     4
  Timeout:   4
  Servers:   lark
  Path:      ucx.ern.sea.com,ern.sea.com,sea.com

 Process

  State:     Enabled

  Transport:
  Domain:
  Retry:
  Timeout:
  Servers:
  Path:
```

Here, host LARK in the current domain is the default name server. To add records to the local hosts database, use the SET HOST command. For example, the following command adds host birdy to the local host database. (See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for more information on using SET commands.)

```
TCPIP> SET HOST birdy /ADDRESS=9.20.208.47
```

To delete server entries from the configuration database, or to add new entries, enter the following command:

```
TCPIP> SET NAME_SERVICE /NOSERVER=LARK /SYSTEM
```

This command modifies the volatile database. To make changes permanent, also enter a SET CONFIGURATION NAME_SERVICE command to add the change to the permanent database. Enter a SHOW CONFIGURATION NAME_SERVICE command to view the results.

5.7.1 Changing the Default Configuration

To add a new server and enable the BIND resolver, enter the following command:

```
TCPIP> SET NAME_SERVICE /SERVER=host /ENABLE /SYSTEM
```

For *host*, specify the host name or IP address of the BIND server or servers that the BIND resolver is to query.

To specify multiple hosts, list them by request preference. The BIND resolver sends the first lookup request to the first host on the list.

If you define a server list and then add a new server with the SET NAME_SERVICE /SERVER command, the new server is added to the end of the list.

SET commands affect the volatile database. To save your changes to the permanent database, use the SET CONFIGURATION commands. The changes you make with the SET CONFIGURATION commands take effect the next time the software starts up. For example:

```
TCPIP> SET CONFIGURATION NAME_SERVICE /SERVER=host /ENABLE

TCPIP> SHOW CONFIGURATION NAME_SERVICE

BIND Resolver Configuration

  Transport:  UDP
  Domain:     ucx.ern.sea.com
  Retry:         4
  Timeout:       4
  Servers:    9.20.208.47, 9.20.208.53
  Path:       No values defined
```

5.7.2 Examples

The following command defines hosts PARROT, SORA, and JACANA as systemwide BIND servers and enables the BIND resolver:

```
PARROT> TCPIP
TCPIP> SET NAME_SERVICE /SERVER=(PARROT,SORA,JACANA) /SYSTEM /ENABLE
```

The following example defines, for the current login session, host OSPREY as the BIND server. As a result, the servers that are defined systemwide are not queried.

```
TCPIP> SET NAME_SERVICE /SERVER=OSPREY
```

5.7.3 Resolver Default Search Behavior

By default, if no search list is defined and the host name as you typed it has no dot (.) in the name, the BIND resolver performs a lookup using the following forms of the host name:

1. The host name, with the default domain appended
2. Just the host name

For example, suppose you enter the command:

```
TCPIP> SHOW HOST OWL
```

Assuming that the default domain is ucx.ern.sea.com, the resolver performs lookups as follows:

1. On the host name and domain owl.ucx.ern.sea.com.
2. If that lookup was unsuccessful, the resolver searches for host owl.

This behavior is different than the resolver lookup behavior in previous releases (UCX BIND Version 4.*x*.). The following section provides more information.

5.7.4 Resolver Search Behavior in Earlier Releases

In previous releases, the resolver performed lookups as follows:

1. Appended the default domain to the host name and performed a lookup.
2. If the previous lookup failed, the resolver removed the leftmost label from the default domain name, appended the result to the host name and performed the lookup.
3. If that lookup failed, the resolver again removed the leftmost label from the default domain name, appended the result to the host name, and performed the lookup.

For each unsuccessful lookup, this procedure was repeated until only two labels remained in the resulting domain name.

If all these attempts failed, the resolver tried just the host name as typed (as long as it contained at least one dot).

For example, suppose you entered the command:

```
TCPIP> SHOW HOST OWL
```

Assuming the default domain was ucx.ern.sea.com, the resolver performed lookups as follows:

1. On owl.ucx.ern.sea.com.
2. If the previous lookup was unsuccessful, the resolver searched for owl.ern.sea.com.
3. If that lookup was unsuccessful, the resolver searched for owl.sea.com.
4. Finally, if the preceding lookup was unsuccessful, the resolver searched for owl.

5.7.5 Setting the Resolver's Domain Search List

The search list is provided to make entering lookup commands easier by not requiring you to type fully qualified domain names. The search list consists of domain names that the resolver uses when performing lookups. By default, the search list consists of only the default domain, which is stored in the TCPIP$CONFIGURATION.DAT file.

You can change the elements in the search list by entering the SET NAME_SERVICE command, as shown in the following example:

```
TCPIP> SET NAME_SERVICE /PATH=(ucx.ern.sea.com,dux.sea.com,mux.ern.sea.com)/SYSTEM
```

For example, if you enter the command:

```
TCPIP> SHOW HOST CANARY
```

The resolver performs lookups in the following sequence:

1. On canary.ucx.ern.sea.com.

2. If the previous lookup was unsuccessful, the resolver searches for canary.dux.sea.com.
3. If that lookup was unsuccessful, the resolver searches for canary.mux.ern.sea.com.
4. If that lookup was unsuccessful, the resolver searches for canary.

In the following output of the SHOW NAME_SERVICE command, the PATH: label shows the search list information entered with the SET NAME_SERVICE /PATH command. This command displays systemwide information and process-specific information (if process-specific information is set).

```
TCPIP> SHOW NAME_SERVICE

BIND Resolver Parameters

Local domain: ucx.ern.sea.com

System

State:     Started, Enabled

Transport: UDP
Domain:    ucx.ern.sea.com
Retry:     4
Timeout:   4
Servers:   ucx, lemng, 16.99.0.10
Path:      ucx.ern.sea.com, dux.ern.sea.com, mux.ern.sea.com

Process

State:     Enabled
Transport:
Domain:
Retry:
Timeout:
Servers:
Path:
$
```

Any additions you make are appended to the end of the search list.

To remove an element from the search list, enter the following command:

```
TCPIP> SET NAME_SERVICE /NOPATH=dux.ern.sea.com /SYSTEM
```

Note

When you execute TCPIP$CONFIG.COM after upgrading from UCX to TCP/IP Services for OpenVMS, the system creates a domain search list that is consistent with the UCX default lookup behavior. TCPIP$CONFIG.COM uses the default domain to create a search list consisting of each parent domain. For example, if the default domain is `ucx.ern.sea.com`, the resulting search list is `ucx.ern.sea.com,ern.sea.com,sea.com`. You can modify the current search list by using the SET CONFIGURATION NAME_SERVER /PATH command.

5.8 Using NSLOOKUP to Query a Name Server

NSLOOKUP is a debugging tool provided with BIND that allows anyone to directly query a name server and retrieve information. Use NSLOOKUP to determine whether your local name server is running correctly or to retrieve information from remote servers.

NSLOOKUP makes direct queries to name servers around the world to obtain DNS information, which includes:

- Host names and addresses on the local domain
- Host names and addresses on remote domains
- Host names that serve as mail exchange (MX records)
- Name servers for a specific zone

5.8.1 Invoking NSLOOKUP

Table 5–12 shows how to start and stop NSLOOKUP.

Table 5–12 Starting and Stopping NSLOOKUP

Task	Command
Run NSLOOKUP.	`$ RUN SYS$SYSTEM:TCPIP$NSLOOKUP.EXE`
Terminate NSLOOKUP from within interactive mode.	`> EXIT`
Terminate the current NSLOOKUP session.	`> Ctrl/Z`

To run NSLOOKUP as a foreign command, enter the following command at the DCL prompt ($), or place the command in your LOGIN.COM file:

```
$ NSLOOKUP :== SYS$SYSTEM:TCPIP$NSLOOKUP.EXE
```

You can then run an interactive NSLOOKUP session by entering:

```
$ NSLOOKUP
>
```

You can put NSLOOKUP `set` commands in an initialization file named SYS$LOGIN:NSLOOKUPINIT.INI. The commands are executed when you start NSLOOKUP.

5.8.2 Obtaining Help

You can obtain help by:

- Entering the following DCL command:

  ```
  $ HELP TCPIP_SERVICES NSLOOKUP
  ```

- Entering one of the following NSLOOKUP commands:

  ```
  > ?
  ```

  or

  ```
  > help
  ```

The following example shows the information available with the NSLOOKUP `help` (?) command:

```
$ NSLOOKUP
Default Server:  condor.lgk.dec.com
Address:  16.99.208.53

> ?

  Information available:

  <host>     About_nslookup        exit       finger     ls         lserver
  root       server     set

Topic?
```

5.8.3 NSLOOKUP Commands

NSLOOKUP interprets unrecognized commands as host names. When using NSLOOKUP, apply these syntax rules:

- The command line must be fewer than 256 characters.
- Commands must be either all uppercase or all lowercase. NSLOOKUP does not accept commands in mixed case.

When NSLOOKUP first starts, you see the name and address of the default BIND server, followed by the NSLOOKUP prompt. In the following example, the default server is `condor.lgk.dec.com`.

```
$ NSLOOKUP
Default Server:  condor.lgk.dec.com
Address:  16.99.208.53

>
```

Table 5–13 lists the NSLOOKUP commands.

Table 5–13 NSLOOKUP Commands

Command	Function

host [*server*]	Looks up information using the current default server or the server you specify. Enter the name of the host for which you need an IP address. For example, the following command searches for www.whitehouse.gov using the default server:
	`> www.whitehouse.gov`
	`Server: condor.lgk.dec.com`
	`Address: 16.99.208.53`
	`Name: www.whitehouse.gov`
	`Addresses: 198.137.240.92, 198.137.240.91`
	The following command uses the server with IP address 128.45.45.40 to look up the host snowy.frozen.weather.com:
	`> snowy.frozen.weather.com 128.45.45.40`

server *domain*	Changes the default server to the domain you specify. The following command changes the default server using the current default server to look up information for the new default server, 128.45.35.40: `> server 128.45.35.40`
lserver *domain*	Changes the default server. The following command changes the default server using the initial default server to find information about 128.45.35.40: `> lserver 128.45.35.40`
root	Changes the default server to the server specified by the root option. `> root`
ls	Lists information about hosts in the domain you specify. The default output contains host names and their IP addresses. The options for ls are listed in Table 5–15. The following example redirects the output from the screen to the file NSLOOKUP.temp in the current directory. `> ls -a frozen.weather.com > NSLOOKUP.temp`
help or ?	Displays a summary of the available commands.
exit	Exits NSLOOKUP.
set	Selects the type of information that NSLOOKUP displays. Table 5–14 lists the available options.

5.8.4 Default Option Values

NSLOOKUP has options that influence the type of information you receive from a query and the way NSLOOKUP behaves. Some of the options take a value and others are Boolean options. The options have default values and can be changed by using the set command.

You obtain a list of the options and their default values by entering the set all command immediately after starting an interactive NSLOOKUP session, as shown in the following example:

```
$ NSLOOKUP
Default Server:  condor.lgk.dec.com       1
Address:  16.99.208.53

> set all                                 2
Default Server:  condor.lgk.dec.com
Address:  16.99.208.53

Set options:
  nodebug             defname         search          recurse  3
  nod2                novc            noignoretc      port=53  4
  querytype=A  5      class=IN        timeout=4       retry=4
  root=a.root-servers.net.
  domain=xyz.prq.dec.com                  6
  srchlist=xyz.prq.dec.com                7

>
```

1 The current default name server. This option is initially set to the name server configured for your local system (client).

2 The set all command displays the current value of each option. When NSLOOKUP starts, it initializes the options with default values. If you enter the set all command immediately after starting up NSLOOKUP, you see the default values.

3 Boolean options are those that have an on/off state. Boolean options that are in the off state are prefixed with the character string NO. For example, nodebug.

4 NSLOOKUP displays options that take a value with an equal sign (=) and the option's current value. For example, port=53.

5 The querytype option specifies the type of information returned on a query. The default query type is A, which returns the host's IP address.

6 The domain option specifies a default domain. When you query with a host name, the default domain is appended to the host name before issuing the query. The defname and search options affect how NSLOOKUP appends the default domain name.

7 The srchlist option specifies 1 to 6 domains to search by default. NSLOOKUP sets the domain option to the first domain listed in the search list.

For a description of other set options, see Table 5–14.

Table 5–14 Options to the NSLOOKUP set Command

Option	Function
all	Displays the current values of the options you can set as well as information about the current default server. For example: > set all

(continued on next page)

Table 5–14 (Cont.) Options to the NSLOOKUP set Command

Option	Function
class=*value*	Changes the query class to one of the following: • IN — The Internet class (default) • CHAOS — The Chaos class • ANY — Wildcard The class specifies the protocol group of the information. You can use the abbreviated form of the keyword cl. The following command tells NSLOOKUP to resolve both internet and chaos class queries (you can enter INTERNET and CHAOS): `> set class=ANY`

querytype	Specifies the type of information you want. For example:
	`> set querytype=A`
	`> set querytype=ANY`
	Valid types include:
	SOA — Start of authority. Marks the beginning of a zone's data and defines parameters that affect the entire zone.
	NS — Name server. Identifies a domain's name server.
	A — Address. Maps a host name to an address.
	ANY — Defines all available resource records for a given name.
	PTR — Pointer. Maps an address to a host name.
	MX — Identifies where to deliver mail for a given domain.
	CNAME — Defines an alias host name.
	HINFO — Host information. Describes a host's hardware and operating system.
	WKS — Well-known service. Advertises network services.

[no]debug	Turns on debugging (default is nodebug). NSLOOKUP displays both detailed information about the packet sent to the server and the answer. You can use the abbreviations nodeb and deb. For example: `> set debug`
[no]d2	Returns all-inclusive debugging information (default is nod2). Displays all the fields of every packet. For example: `> set d2`
recurse	Tells the BIND server to contact other servers if it does not have the information you want. The servers carry out a complete (recursive) resolution for each query. For example: `> set recurse`
retry	Number of times that NSLOOKUP attempts to contact a BIND server if repeated tries fail. For example: `> set retry=8`

(continued on next page)

Table 5–14 (Cont.) Options to the NSLOOKUP set Command

Option	Function
timeout	Length of time (in seconds) to wait for a reply from each attempt. For example: `> set timeout=9`
root=*value*	Changes the root server. For example, the following command changes the root server to ns.nasa.gov. `> set root=ns.nasa.gov`
ignoretc	Tells NSLOOKUP to ignore packet truncation errors. For example: `> set ignoretc`
domain *name*	Changes the default domain to the domain you specify. How the default domain name is appended to lookup requests depends on the settings of the defname and search options. The domain search list contains the parents of the default domain if it has at least two components in its name. The default value is set in the TCP/IP configuration database. To specify the default, type the abbreviation do. For example, if the default domain is CC.Berkeley.EDU, the search list is CC.Berkeley.EDU and Berkeley.EDU.
srchlist	If set, NSLOOKUP appends each of the domain names specified in the srchlist option to an unqualified host name and performs a query until an answer is received.
srchlist=*names*	Changes the default domain name to the first name you specify and the domain search list to all the names you specify. Specify a maximum of six names separated by slashes (/). In the following example, the command sets the default domain to lcs.MIT.EDU and the search list to the three specified domains. The command overrides the default domain name and associated search list for the set domain command. `> set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU` The default is the domain name specified in the TCP/IP configuration database. The abbreviated form of the command is srchl.
[no]defname	Tells NSLOOKUP to append a default domain name to a not fully qualified[1] DNS name lookup request. The abbreviated form is [no]def. For example, an NSLOOKUP query for the host rainy becomes rainy.cc.berkeley.edu.
[no]search	Tells NSLOOKUP to append the domain names in the search list to a not fully qualified domain name[1] lookup request. The default is search. The abbreviated form is [no]sea.

[1]A fully qualified domain name is a name that ends with a trailing period, as in *host.domain.*

5.8.5 Query Types

You can change the type of information you receive from a query. The default query type is A. Table 5–14 lists the different types of query information.

5.8.5.1 A Query Type

This is the default NSLOOKUP query type. It returns the name and IP address of a host. The following NSLOOKUP session shows a query for the host `apple`. The query to the server `condor.lgk.dec.com` is successful, and the server returns the IP address 16.99.208.10.

```
$ NSLOOKUP
Default Server:  condor.lgk.dec.com
Address:  16.99.208.53

> apple
Server:  condor.lgk.dec.com
Address:  16.99.208.53

Name:    apple.lgk.dec.com
Address:  16.99.208.10
>
```

If you enter a domain name without a trailing period, NSLOOKUP appends the default domain to the name. You can change the default domain with the `set domain` or `set srchlist` commands.

To look up a host not in the current domain, append a period to the name, as shown in the following example:

```
$ NSLOOKUP apple.koz.dec.com.
```

5.8.5.2 PTR Query Type

To obtain the host name for an IP address, change the query type to PTR and enter the IP address, as shown in the following example:

```
> set type=ptr
> 16.99.208.189
Server:  condor.lgk.dec.com
Address:  16.99.208.53

Name:    dove.lgk.dec.com
Address:  16.99.208.189
```

You can also use the PTR query type to obtain more information about a domain, as as shown in the following example:

```
> lgk.dec.com
Server:  condor.lgk.dec.com
Address:  16.99.208.53

lgk.dec.com
        origin = condor.lgk.dec.com
        mail addr = postmaster.lgk.dec.com
        serial = 1998101948
        refresh = 3600 (1H)
        retry   = 300 (5M)
        expire  = 604800 (1W)
        minimum ttl = 43200 (12H)
>
```

5.8.5.3 MX Query Type

To obtain information about mail exchange records, set the query type to MX and enter a domain. The output tells you which hosts handle mail for the specified domain, as shown in the following example.

```
> set type=mx
> lgk.dec.com
Server:  condor.lgk.dec.com
Address:  16.99.208.53

lgk.sea.com preference = 200, mail exchanger = crl.sea.com
lgk.sea.com preference = 50, mail exchanger = collie.lgk.sea.com
lgk.sea.com preference = 100, mail exchanger = mail13.digital.com
lgk.sea.com preference = 100, mail exchanger = mail11.digital.com
lgk.sea.com preference = 200, mail exchanger = mail2.digital.com
lgk.sea.com nameserver = collie.lgk.sea.com
lgk.sea.com nameserver = condor.lgk.sea.com
lgk.sea.com nameserver = hageln.lgk.sea.com
crl.sea.com     internet address = 192.58.206.2
collie.lgk.sea.com  internet address = 16.99.208.100
mail13.digital.com      internet address = 192.208.46.30
mail2.digital.com       internet address = 204.123.2.56
condor.lgk.sea.com  internet address = 16.99.208.53
hageln.lgk.sea.com  internet address = 16.99.208.10
```

5.8.5.4 SOA Query Type

This query type returns the domain's start-of-authority information.

```
> set type=soa
> microsoft.com
Server:  condor.lgk.sea.com
Address:  16.99.208.53

microsoft.com
        origin = dns1.microsoft.com
        mail addr = msnhst.microsoft.com
        serial = 1998101204
        refresh = 7200 (2H)
        retry   = 1800 (30M)
        expire  = 2592000 (4w2d)
        minimum ttl = 86400 (1D)
microsoft.com   nameserver = dns3.nwnet.net
microsoft.com   nameserver = dns4.nwnet.net
microsoft.com   nameserver = dns1.microsoft.com
microsoft.com   nameserver = dns2.microsoft.com
microsoft.com   nameserver = dns1.moswest.msn.net
microsoft.com   nameserver = dns2.moswest.msn.net
dns3.nwnet.net  internet address = 192.220.250.7
dns4.nwnet.net  internet address = 192.220.251.7
dns1.microsoft.com      internet address = 131.107.1.7
dns2.microsoft.com      internet address = 131.107.1.240
```

5.8.5.5 NS Query Type

To obtain information about the name servers for a particular zone, set the query type to NS and then enter the zone you want. The following example shows the name servers for the microsoft.com zone.

```
> set type=ns
> microsoft.com
Server:  condor.lgk.sea.com
Address:  16.99.208.53
```

```
Non-authoritative answer:
microsoft.com   nameserver = dns2.microsoft.com
microsoft.com   nameserver = dns1.moswest.msn.net
microsoft.com   nameserver = dns2.moswest.msn.net
microsoft.com   nameserver = dns3.nwnet.net
microsoft.com   nameserver = dns4.nwnet.net
microsoft.com   nameserver = dns1.microsoft.com

Authoritative answers can be found from:
dns2.microsoft.com      internet address = 131.107.1.240
dns3.nwnet.net  internet address = 192.220.250.7
dns4.nwnet.net  internet address = 192.220.251.7
dns1.microsoft.com      internet address = 131.107.1.7
>
```

5.8.6 Changing the Default Server

If you want to use another name server as your default server, use the server command.

```
$ NSLOOKUP
Default Server:  condor.klg.sea.com
Address:  16.99.208.53

> server ns01.koz.sea.com
Default Server:  ns01.koz.sea.com
Address:  16.99.9.20
```

If for some reason the default server is not responding, you can always use the lserver command to change the default server. The lserver command uses the initial default name server to look up the IP address of the new server.

```
> lserver collie.klg.sea.com
Default Server:  collie.klg.sea.com
Address:  16.99.208.10
```

Or, if you already know the IP address of the new server, you can use the server command to reset the default server.

```
>server 16.99.208.10

> server 16.99.99.226
Default Server:  beagel.zok.sea.com
Address:  16.99.99.226
```

5.8.7 Listing Domain Information

The ls command lists information about a domain. This command is useful for:

- Determining the number of hosts within a domain
- Host names and their IP addresses
- Troubleshooting DNS problems

Table 5–15 describes options to the ls command.

Table 5–15 Options to the NSLOOKUP ls Command

Option	Function
-a	Lists aliases of hosts in the domain (CNAME entries).
-d	Lists all the entries in the domain.
-h	Lists CPU and operating system information for the domain (HINFO entries).
-m	Lists mail exchange (MX) entries in the domain.
-s	Lists well-known services (WKS) in the domain.
-t	Lists a specified entry type.

The following example shows the use of the ls command to obtain address records for all hosts within a zone.

```
> ls -t a lgk.sea.com
[condor.lgk.sea.com]
@                       12H IN A        16.99.208.208
dhcp-253                12H IN A        16.99.208.253
ucxv4a                  12H IN A        16.99.208.129
beavis                  12H IN A        16.99.208.90
boxmor                  12H IN A        16.99.208.30
kempo                   12H IN A        16.99.208.47
pacnet                  12H IN A        16.99.208.84
kwai                    12H IN A        16.99.208.63
alxica                  12H IN A        16.99.9.37
ppponvms                12H IN A        16.99.208.104
a71kt                   12H IN A        16.99.208.142
peteathome              12H IN A        16.99.208.101
larisa                  12H IN A        16.99.208.49
pigdog                  12H IN A        16.99.208.140
ntruder                 12H IN A        16.99.208.110
```

In the following example, the ls command displays alias records for hosts within the lgk.sea.com domain.

```
> ls -a lgk.sea.com
[condor.lgk.sea.com]
$ORIGIN LGK.SEA.COM.
celics                  12H IN CNAME    celtics
news                    12H IN CNAME    nntpd.KLG.SEA.COM.
tiger                   12H IN CNAME    ntruder
console                 12H IN CNAME    bblts.KLG.SEA.COM.
deebug                  12H IN CNAME    dot
ayla                    12H IN CNAME    ayla.KLG.SEA.COM.
cscibm                  12H IN CNAME    cscibm.KLG.SEA.COM.
>
```

Using the -m option obtains the MX records for hosts within the lgk.sea.com domain, as shown in the following example.

```
> ls -m lgk.sea.com
brigit                  12H IN MX       10 brigit
                        12H IN MX       100 mail1.digital.com.
                        12H IN MX       100 mail2.digital.com.
                        12H IN MX       200 crl.SEA.com.
piglet                  12H IN MX       10 piglet
                        12H IN MX       100 mail1.digital.com.
                        12H IN MX       100 mail2.digital.com.
                        12H IN MX       200 crl.SEA.com.
tieta                   12H IN MX       10 tieta
                        12H IN MX       100 mail1.digital.com.
                        12H IN MX       100 mail2.digital.com.
                        12H IN MX       200 crl.SEA.com.
sherry                  12H IN MX       10 sherry
                        12H IN MX       100 mail1.digital.com.
                        12H IN MX       100 mail2.digital.com.
                        12H IN MX       200 crl.SEA.com.
```

In the following example, using the `-S` option displays the well-known services for a domain.

```
> ls -s lgk.sea.com
[condor.lgk.sea.com]
WKStesthave             12H IN WKS      16.99.208.255 21 ( )
WKStesthavenot          12H IN WKS      16.99.208.255 255 ( )
WKStestnumbers          12H IN WKS      16.99.208.255 255 ( 21 23 )
```

You can redirect the output from this command to a file. This method is helpful when the domain consists of a large number of hosts. Once the file is created, you can look at its contents with the DCL command TYPE. For example:

```
> ls -t a klg.sea.com > systems.txt
[condor.klg.sea.com]
#############
Received 932 answers (0 records).
```

```
$ TYPE SYSTEMS.TXT
> ls -t a klg.sea.com
[condor.klg.sea.com]
$ORIGIN KLG.SEA.COM.
@                       12H IN A        16.99.208.208
dhcp-253                12H IN A        16.99.208.253
ucxv4a                  12H IN A        16.99.208.129
beavis                  12H IN A        16.99.208.90
boxmor                  12H IN A        16.99.208.30
kempo                   12H IN A        16.99.208.47
pacnet                  12H IN A        16.99.208.84
kwai                    12H IN A        16.99.208.63
alxica                  12H IN A        16.99.9.37
ppponvms                12H IN A        16.99.208.104
```

5.9 Solving Bind Server Problems

To solve BIND server problems, refer to the following sections:

- Section 5.9.1, Server Not Responding
- Section 5.9.2, Serial Number Mismatch

5.9.1 Server Not Responding

A missing client name in the BIND server's database files results in lack of service to that client. If records that point to the name servers (NS records) in a domain are missing from your server's database files, you might see the following messages:

```
%TCPIP-W-BIND_NOSERVNAM, Server with address 199.85.8.8 is not responding
%TCPIP-E-BIND_NOSERVERS, Default servers are not available
%TCPIP-W-NORECORD, Information not found
-TCPIP-E-BIND_NOSERVERS, Default servers are not available
```

When the CONVERT/ULTRIX BIND /DOMAIN command creates the .DB files from the hosts database, it cannot detect the existence of or the names of name servers in a domain. Therefore, it does not add NS records for the name servers to the .DB files.

To solve the problem, follow these steps:

1. Stop the BIND server.
2. Manually add NS records for the missing names.
3. Update the start-of-authority (SOA) records by incrementing the serial number.
4. Restart the BIND server.

5.9.2 Serial Number Mismatch

The `serial number mismatch log=`*`xxx`*`, zone=`*`xxx`* message indicates that the BIND server was unsuccessful in an attempt to load a particular dynamic update from the update log file. This can occur because the update is old and no longer valid. The serial number for the zone has since been incremented and is no longer in synchronization with the serial number attached to the logged update.

6
Using DNS to Balance Work Load

This chapter describes how to use DNS to balance the network traffic on a multihomed host or on network servers when you have multiple systems providing the same network service.

TCP/IP Services provides two methods for balancing work load using DNS:

- Load sharing using the default DNS method of round-robin scheduling.
- Load balancing using the TCP/IP Services load broker. Load broker is a configurable, calculated, load-balancing mechanism for distributing the work load among DNS cluster members.

This chapter discusses how to use DNS to balance server work load and includes the following topics:

- DNS clusters (Section 6.1)
- Round-robin scheduling (Section 6.2)
- Load broker concepts (Section 6.3)
- Load broker startup and shutdown (Section 6.4)
- Configuring the load broker (Section 6.5)
- Metric server startup and shutdown (Section 6.6)
- Solving load broker problems (Section 6.7)

6.1 DNS Clusters

TCP/IP Services defines the term **DNS cluster** to refer to several A resource records for a single host name. This could be the A resource records for a multihomed host or the A resource records for one or more servers which are to share a work load.

6.2 Round-Robin Scheduling

Round-robin scheduling is the default load-sharing method used by a DNS server. If multiple resource records satisfy a query, the BIND server returns them each time in a round-robin order. The round-robin scheme is a simple rotation where client requests are passed from one cluster member to the next. The round-robin scheme is also useful for MX records to share mail loads among multiple equivalent gateways of the same MX preference.

Unlike the traditional load-balancing method, round-robin does not take into account the current work load on the DNS cluster members and does not know whether these hosts are up or down.

The following example demonstrates how round-robin load sharing works.

In the example, the DNS cluster alias is defined as robin. When the DNS server receives queries for robin, it shuffles the A resource records in a round-robin manner.

```
;
; TCP/IP DNS cluster load sharing - round robin method
;    DNS cluster alias:   "robin"
robin                           IN      A       9.20.208.47
                                IN      A       9.20.208.30
                                IN      A       9.20.208.72
;
birdy                           IN      A       9.20.208.47
seagull                         IN      A       9.20.208.30
owl                             IN      A       9.20.208.72
;
```

A user enters the TELNET command, specifying the DNS cluster alias ROBIN. The first query to the DNS name server results in the following TELNET session:

```
$ TELNET ROBIN
%TELNET-I-TRYING, Trying ... 9.20.208.47
%TELNET-I-SESSION, Session 01, host birdy, port 23
-TELNET-I-ESCAPE, Escape character is ^]
```

The TELNET client connects to host birdy at IP address 9.20.208.47, the first resource record in the list.

The second query to the name server results in the following TELNET session:

```
$ TELNET ROBIN
%TELNET-I-TRYING, Trying ... 9.20.208.30
%TELNET-I-SESSION, Session 01, host seagull, port 23
-TELNET-I-ESCAPE, Escape character is ^]
```

The TELNET client connects to host seagull at IP address 9.20.208.30, the next resource record in the list.

The third query to the name server results in the following TELNET session:

```
$ TELNET ROBIN
%TELNET-I-TRYING, Trying ... 9.20.208.72
%TELNET-I-SESSION, Session 01, host owl, port 23
-TELNET-I-ESCAPE, Escape character is ^]
```

TELNET connects to host owl at IP address 9.20.208.72, the next resource record in the list.

The fourth query to the name server results in the following TELNET session:

```
$ TELNET ROBIN
%TELNET-I-TRYING, Trying ... 9.20.208.47
%TELNET-I-SESSION, Session 01, host birdy, port 23
-TELNET-I-ESCAPE, Escape character is ^]
```

TELNET again connects to host birdy at IP address 9.20.208.47. This is the start of the cycle repeating. The cycle repeats for the subsequent queries.

The SHOW HOST display for this DNS name server shows the shuffling effect in greater detail. Notice that the output displays the cluster alias name for each host involved in round-robin scheduling.

```
TCPIP> SHOW HOST ROBIN
     BIND database

Server:   9.20.208.72 owl.ucx.ern.sea.com

Host address    Host name

9.20.208.47 robin.ucx.ern.sea.com
9.20.208.30 robin.ucx.ern.sea.com
9.20.208.72 robin.ucx.ern.sea.com

TCPIP> SHOW HOST ROBIN
     BIND database

Server:   9.20.208.72 owl.ucx.ern.sea.com

Host address    Host name

9.20.208.30 robin.ucx.ern.sea.com
9.20.208.72 robin.ucx.ern.sea.com
9.20.208.47 robin.ucx.ern.sea.com

TCPIP>  SHOW HOST ROBIN
     BIND database

Server:   9.20.208.72 owl.ucx.ern.sea.com

Host address    Host name

9.20.208.72 robin.ucx.ern.sea.com
9.20.208.47 robin.ucx.ern.sea.com
9.20.208.30 robin.ucx.ern.sea.com

TCPIP> SHOW HOST ROBIN
     BIND database

Server:   9.20.208.72 owl.ucx.ern.sea.com

Host address    Host name

9.20.208.47 robin.ucx.ern.sea.com
9.20.208.30 robin.ucx.ern.sea.com
9.20.208.72 robin.ucx.ern.sea.com
```

6.2.1 Disabling Round-Robin Scheduling

If you want to disable round-robin scheduling, use the DCL command DEFINE to define the following logical:

```
$ DEFINE/SYSTEM TCPIP$BIND_ROUND_ROBIN_OFF
```

6.3 Load Broker Concepts

TCP/IP Services provides a configurable, calculated, load-balancing mechanism called the load broker for distributing the load across systems in a DNS cluster.

Unlike round-robin scheduling (the default method used by most DNS name servers), the load broker takes into account the load on all DNS cluster participants. The load broker polls DNS cluster members and updates the DNS namespace accordingly.

6.3.1 How the Load Broker Works

When the load broker starts, it reads its configuration file and starts polling DNS cluster members. The load broker exchanges messages with DNS cluster members that run the metric server. The metric server (Section 6.3.2) calculates the current rating and reports it when polled by the load broker. Periodically, the load broker sorts the list of addresses based on metric rating reports, drops the systems that are not responding after being polled three times, and takes a subset of the list and compares it to the name server information. To do the comparison, the load broker sends a host lookup request to the specified name server. If the lists are the same, the load broker does not make any change. If the lists are different, the load broker updates the name server data by sending a dynamic update request to the specified name server.

The name server uses round-robin scheduling to further balance the load across the members of a DNS cluster. So every consecutive request for translating the DNS cluster name results in a list being returned, rotated by one.

The `dns-ttl` value stored in the load broker configuration file governs how long the record is to be cached by other name servers. If some intermediate name server caches the "A" resource records for a given DNS cluster name, it caches it for the period of time defined by the `dns-ttl` value. The default `dns-ttl` value is 45 seconds. If less time is required, you can set `dns-ttl` to a smaller value. To suppress any caching, you can set the `dns-ttl` to 0.

The `dns-refresh` time specifies how often the DNS information for a given DNS cluster is refreshed. The default is 30 seconds. If you want to quickly pick up changes in the system load (reported by metric servers), set `dns-refresh` to a smaller number.

If the load broker has not received a response from a metric server after three polling intervals, the load broker marks the address for removal from the DNS alias. This removal will occur at the next `dns-refresh` interval.

For earliest possible detection of this failure, the `polling-interval` should be set to a value that is less than one-third of the value specified as the `dns-refresh` period.

```
polling-interval  < (dns-refresh) / 3
```

The `masters` list specifies the name server to use for dynamic updates. This must point to the master name server. The name server must be set up to allow dynamic updates from the system that runs the load broker. For how to configure dynamic updates, see Section 5.3.6.

TCP/IP Services supports dynamic updating of only one master server in a DNS cluster environment.

6.3.2 How the Metric Server Calculates Load

The metric server calculates the current load on a DNS cluster host by using the following equation:

```
rating = availablity + workload - penalty
```

In the equation, the variables are calculated by:

- Availability

Availability is calculated using the IJOBLIM system parameters and the SDA global reference variable IJOBCNT in the following equation:

```
availablity = (20*(IJOBLIM-IJOBCNT))/IJOBLIM
```

- Workload

  One consideration in the work load calculation is the system manager's estimate of the host's relative CPU power specified by the system logical TCPIP$METRIC_CPU_RATING.

  To set a CPU power value, use the DCL command DEFINE to define the system logical name TCPIP$METRIC_CPU_RATING with a value. The CPU rating value can range from 1 (the lowest CPU power) to 100 (the highest CPU power). If a value is specified, the value is used instead of the term (min(235,IJOBLIM) in the following equation.

  ```
  workload = (min(235,IJOBLIM)*100)/(100+load_average)
  ```

  When you set the logical value to 0, or if you do not define TCPIP$METRIC_CPU_RATING, the metric server uses the value of the system parameter IJOBLIM to calculate work load.

  load_average is an average of the current CPU load taken every second. It is calculated by using 97.9% of the previous CPU load and 2.1% of the current CPU load value.

- Penalty

  The metric server uses the FREEGOAL system parameter and the SDA global reference variable FREECNT to calculate an available memory penalty.

  ```
  penalty = 40*((FREEGOAL+2048-FREECNT)/(FREEGOAL+2048))
  ```

  The value of penalty is subtracted from the rating only if the value is positive. If the value of FREECNT is high enough, the value of penalty is not applied.

6.4 Load Broker Startup and Shutdown

The load broker can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$LBROKER_STARTUP.COM allows you to start up the load broker service.
- SYS$STARTUP:TCPIP$LBROKER_SHUTDOWN.COM allows you to shut down the load broker service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$LBROKER_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the load broker is started.
- SYS$STARTUP:TCPIP$LBROKER_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the load broker is shut down.

6.5 Configuring the Load Broker

To configure the load broker, edit the file TCPIP$LBROKER_CONF.TEMPLATE located in SYS$SYSDEVICE:[TCPIP$LD_BKR], then rename the file to TCPIP$LBROKER.CONF.

After making changes to TCPIP$LBROKER.CONF, restart the load broker by running TCPIP$CONFIG, or by using the shutdown and startup procedures.

The load broker configuration file can contain one or more DNS cluster statements in the following format:

```
cluster "clustername.domain.com"
{
 [dns-ttl nn;]
 [dns-refresh nn;]
 masters {ip_address};
 [polling-interval nn;]
 [max-members nn;]
 members {ip_address};
 failover {ip_address};
};
```

Table 6–1 describes the valid cluster statements.

Table 6–1 Valid Cluster Statements

Statement	Description
`members`	Specifies the IP address for each DNS cluster member.
`failover`	Specifies the address of the host to use if all other members are down.
`masters`	Specifies the IP addresses of authoritative name servers.
`dns-ttl`	Specifies the time to live for a given record. The value you provide governs how long the record is to be cached by other name servers. If some intermediate name servers cache A resource records for a given DNS cluster name, they cache it for the period specified by `dns-ttl` for the resource record. The default value is 45 seconds.
`dns-refresh`	Specifies how often the DNS information for a given DNS cluster name is refreshed. The default is 30 seconds. The value of this field should be set relative to to the value of `polling-interval`. The `dns-refresh` value should be smaller than the `polling-interval` value. It is unproductive to refresh more often then you poll.
`polling-interval`	Specifies the length of time between polls to cluster members. The default is 30 seconds.
`max-members`	Specifies the maximum number of IP addresses to be returned to the name server in each dynamic update. For effective load balancing, this number should be between one-third and one-half the number of participating DNS cluster members.

The following sample is a configuration of the load broker that load balances the DNS cluster named WWW.TCPIP.ERN.SEA.COM.

```
cluster "www.tcpip.ern.sea.com"
{
 dns-ttl   45;
 dns-refresh  30;
 masters {
  9.20.208.53;
 };
 polling-interval  9;
 max-members   3;
 members {
  9.20.208.100;
  9.20.208.53;
  9.20.208.54;
  9.20.208.80;
  9.20.208.129;
  9.20.208.130;
 };
 failover 16.20.208.200;
};
```

To retain your UCX Version 4.*x* DNS cluster load-balancing configuration:

1. Enter the CONVERT/CONFIGURATION BIND/CLUSTER command, as shown in the following example:

   ```
   TCPIP> CONVERT/CONFIGURATION BIND -
   _TCPIP> /CLUSTER=SYS$SYSDEVICE:[TCPIP$LD_BKR]TCPIP$LBROKER.CONF
   ```

   The output from this command is a TCPIP$LBROKER.CONF file containing your basic configuration.

2. Edit the TCPIP$LBROKER.CONF file to produce a complete configuration file.

6.5.1 Enabling the Load Broker

To enable DNS cluster load balancing, complete the following tasks:

1. Ensure that all hosts in the DNS cluster are running TCP/IP Services.
2. Configure the load broker (see Section 6.5).
3. Configure the BIND name server that is authoritative for the DNS cluster to allow dynamic updates from the host on which the load broker is running. For how to configure dynamic updates, see Section 5.3.6.
4. Ensure TCP/IP connectivity between the DNS cluster members and the load broker.
5. Enable the metric server on each member of the DNS cluster:
   a. Run the following command procedure:

      ```
      $ @SYS$MANAGER:TCPIP$CONFIG
      ```

   b. On the TCPIP$CONFIG Server Components Configuration menu, select option 8:

      ```
      8 -- METRIC.
      ```

   c. On the Metric configuration display, select option 2:

      ```
      2 -- Start service on this node.
      ```

Review the following guidelines:

- DNS cluster hosts and clients are not required to be on the same bridged LAN.

- The number of DNS cluster member hosts is limited to 32.
- A BIND name server can also be a DNS cluster member host.
- The authoritative name server can run any BIND name server that supports BIND 8.1.1 or later or that supports dynamic updates.

6.5.2 Load Broker Logical Names

Table 6–2 describes the load broker's logical names. Define these logical names with the /SYSTEM qualifier, and restart the load broker server to make the changes take effect.

Table 6–2 Load Broker Logical Names

Logical Name	Description
TCPIP$LBROKER_LOG_LEVEL *value*	Turns on diagnostics and writes them to the TCPIP$LBROKER_RUN.LOG located in SYS$SYSDEVICE:[TCPIP$LD_BKR]. Valid values are 1 and 2 (2 provides more detailed diagnostics).

6.5.3 Metric Server Logical Names

Table 6–3 describes the metric server's logical names. Define these logical names with the /SYSTEM qualifier. The metric server detects the change and dynamically updates the current enviroment.

Table 6–3 Metric Server Logical Names

Logical Name	Description
TCPIP$METRIC_CPU_RATING *value*	Sets a bias value that represents your estimate of the relative CPU power. Valid values range from 1 (lowest CPU power) to 100 (highest CPU power). Use a value of 0 (zero) to specify the default (The value of the system parameter IJOBLIM is used).
TCPIP$METRIC_COMPUTE_INTERVAL *value*	Specifies how often the metric server computes the rating. Valid value (in seconds) is a number from 1 to 300. The default is 10 seconds.
TCPIP$METRIC_LOG_LEVEL *value*	Turns on diagnostics logged to the file TCPIP$METRIC_RUN.LOG located in SYS$SPECIFIC:[TCPIP$METRIC]. Valid values are 1 or 2 (2 provides more detailed diagnostics).

6.6 Metric Server Startup and Shutdown

The metric server starts up automatically at system startup time if the service was previously enabled and can be shut down and started independently.

The following files are provided:

- SYS$STARTUP:TCPIP$METRIC_STARTUP.COM allows you to start up the metric service.

- SYS$STARTUP:TCPIP$METRIC_SHUTDOWN.COM allows you to shut down the metric service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$METRIC_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the metric service is started.
- SYS$STARTUP:TCPIP$METRIC_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the metric service is shut down.

6.7 Solving Load Broker Problems

TCP/IP Services provides the following tools to assist in solving load broker problems:

- The Metric View utility, to display metric information regarding DNS cluster members.
- Diagnostic log files.

6.7.1 Metric View Utility

Two images in the TCP/IP Services software distribution kit perform the metric functions:

- TCPIP$METRIC.EXE — does the computing
- TCPIP$METRICVIEW.EXE — reads the metric ratings

The Metric View utility, SYS$COMMON:[SYSEXE]TCPIP$METRICVIEW.EXE, displays the metric rating of the member hosts in the TCP/IP DNS cluster.

To run Metric View, enter the following command:

```
$ RUN SYS$SYSTEM: TCPIP$METRICVIEW
Host                                                    Rating
----                                                    ------
10.10.2.11      rufus.lkg.dec.com                         47
10.10.2.255     peach.lkg.dec.com                         51
```

6.7.2 Viewing Diagnostic Messages

If you define the logical TCPIP$METRIC_LOG_LEVEL, the METRIC server writes diagnostic messages to the TCPIP$METRIC_RUN.LOG file. If you are experiencing problems with the metric server, define TCPIP$METRIC_LOG_LEVEL and after a period of operation, review the messages in the TCPIP$METRIC_RUN.LOG file for an indication of what the problem could be. See Section 6.5.3 for a description of the logical.

Part 3

Configuring Services

Part 3 describes how to set up and manage the Dynamic Host Configuration Protocol (DHCP), the Bootstrap Protocol (BOOTP), the Trivial File Transport Protocol (TFTP), the Portmapper service, the Network Time Protocol (NTP), and the Simple Network Management Protocol (SNMP). The chapters in this part include the following:

- Chapter 7, Configuring the DHCP Server, describes how to configure the DHCP server so you can centralize the configuration and maintenance of the IP address space.
- Chapter 8, Configuring the DHCP Client, describes how to set up the system as a DHCP client.
- Chapter 9, Configuring BOOTP, describes how to configure the BOOTP server so your host can answer bootstrap requests from diskless workstations and other network devices.
- Chapter 10, Configuring TFTP, describes how to configure the TFTP server to handle file transfers from diskless clients and remote systems.
- Chapter 11, describes how to configure the portmapper service, a service that registers server programs written using RPCs (remote procedure calls). You must run the portmapper service if you intend to run NFS or any customer-developed RPC programs.
- Chapter 12, Configuring and Managing NTP, describes how to configure and manage the NTP (Network Time Protocol), allowing your host to synchronize its time with that of other internet hosts also running NTP.
- Chapter 13, Configuring SNMP, describes how to configure your host so it can answer SNMP (Simple Network Management Protocol) requests from remote SNMP management stations.

7

Configuring the DHCP Server

Dynamic Host Configuration Protocol (DHCP), a superset of the Bootstrap Protocol (BOOTP), provides a centralized approach to the configuration and maintenance of IP address space. It allows the system manager to configure various clients on a network from a single location.

DHCP allocates temporary or permanent IP addresses from an address pool to client hosts on the network. DHCP can also configure client parameters such as default gateway parameters, domain name server parameters, and subnet masks for each host running a DHCP client.

This chapter reviews key DHCP and BOOTP concepts and also describes:

- DHCP server components (Section 7.2)
- DHCP server startup and shutdown(Section 7.3)
- Configuring DHCP server (Section 7.4)
- Using DHCP GUI to configure DHCP (Section 7.5)
- Configuring DHCP/BOOTP IP addressing (Section 7.6)
- Configuring DHCP manually (Section 7.7)
- Supporting utilities (Section 7.8)
- Solving DHCP server problems (Section 7.9)

7.1 Key Concepts

With DHCP, system managers can centralize TCP/IP network configurations and management tasks involved with network connections. DHCP makes network administration easier by allowing:

- Consistent application of network parameters, such as subnet masks and default routers, to all hosts on a network
- Support for both DHCP and BOOTP clients
- Static (permanent) mapping of hardware addresses to IP addresses
- Dynamic (temporary) mapping of hardware addresses to IP addresses, where the client leases the IP address for a defined length of time

In addition, the TCP/IP Services implementation of DHCP includes support for DHCP server failover in a OpenVMS Cluster environment.

The DHCP protocol is a superset of BOOTP. In addition to the BOOTP functionality, DHCP offers robust configuration services, including IP addresses, subnet masks, and default gateways.

Based on the BOOTP functionality, DHCP is built on the client/server model:

- The **DHCP server** is a host that provides initialization parameters.
- The **DHCP client** is a host that requests initialization parameters from a DHCP server. A router cannot be a DHCP client.

7.1.1 How DHCP Operates

DHCP consists of two components:

- A mechanism for allocating network addresses to clients
- A set of rules for delivering client-specific configuration parameters from a DHCP server to a client

DHCP operates as follows:

- When a DHCP client boots, it broadcasts a DHCP request, asking that any DHCP server on the network provide it with an IP address and configuration parameters.
- A DHCP server on the network authorized to configure this client sends the client a reply that offers an IP address.
- When the client receives the offer, it can accept it or wait for other offers from other servers on the network.
- Once the client accepts an offer, it sends an acceptance message to the server.
- When the server receives the acceptance message, it sends an acknowledgment with the offered IP address and any other configuration parameters that the client requested. (The server only responds to specific client requests; it does not impose any parameters on the client.)
- If the dynamic address allocation method is used, the IP address offered to the client has a specific lease time that determines how long the IP address is valid.

During the lifetime of the lease, the client repeatedly asks the server to renew. If the client chooses not to renew, the lease expires.

Once the lease expires, the IP address can be recycled and given to another client. When the client reboots, it can be given the old address if available or assigned a new address.

For more information about how DHCP operates, see RFC 2131 and RFC 1534.

7.1.2 How DHCP Allocates IP Addresses

With TCP/IP Services, DHCP uses the dynamic and static IP address-mapping methods outlined in Table 7–1 to service DHCP and BOOTP-only client requests.

Table 7–1 DHCP IP Address Allocation Methods

Method	Applicable Client	Description
Dynamic	DHCP and BOOTP	The DHCP server assigns an IP address from an address pool to a client for a specified amount of time (or until the client explicitly relinquishes the address). Addresses no longer needed by clients can be reused. Use dynamic allocation when: • Clients plan to be connected to the network only temporarily. • You have a limited pool of IP addresses that must be shared among clients that do not need permanent IP addresses. • IP addresses are scarce, and you need to reclaim retired addresses so you can assign them the new clients being permanently connected to the network. For BOOTP clients, DHCP assigns dynamic IP addresses from the address pool and stores the addresses in the lease database by assigning each lease a time of infinity.
Static	DHCP and BOOTP	The system manager manually assigns (in the DHCPCAP. file) an IP address to a client and uses DHCP to pass the assigned address to the client. Use static allocation in an error-prone environment where it is desirable to manage IP address assignment outside of the DHCP functionality.
Finite	BOOTP-only	The DHCP server assigns an IP address from the pool to the BOOTP client and defines a lease time based on certain parameters you define in the SERVER.PCY file. When the lease expires, the DHCP server pings the IP address. If the server receives a reply, it extends the lease and does not offer the address to a new client. If not, the address is free and can be assigned to a new client.

Section 7.5 explains how to configure the different types of addressing for clients on your network.

The typical network uses a combination of static and dynamic DHCP addressing. As the local system manager or network administrator, you can apply any of the IP addressing methods as appropriate for your specific policies and environment.

7.1.3 Relationship Between DHCP and BOOTP

From the client's perspective, DHCP is an extension of the BOOTP functionality. DHCP allows existing BOOTP clients to operate with DHCP servers without having to change the client's initialization software.

Based on the format of BOOTP messages, the DHCP message format does the following:

- Captures the BOOTP relay agents and eliminates the need to have a DHCP server on each physical network segment.
- Allows existing BOOTP clients to operate with DHCP servers.

Messages that include a DHCP message-type option are assumed to have been sent by a DHCP client. Messages without the DHCP message-type option are assumed to have been sent by a BOOTP client.

However, DHCP improves the BOOTP-only functionality in the following ways:

- DHCP allows the serial reassignment of network addresses to different clients by assigning a network address for a finite lease period.
- DHCP allows clients to acquire all of the IP configuration parameters they need to operate.

7.1.4 Client ID

With BOOTP, a client is identified by its unique media access control (MAC) address that is associated with the network adapter card.

DHCP uses a client identifier (ID) to uniquely identify the client and associate it with a lease: The client creates the client ID from one of the following types of addresses:

- The MAC address.
- A variation of the MAC address. For example, Windows 95 and Windows NT clients create the client ID by prepending the hardware type to the hardware address.

If the client does not include a client ID in the request, the server uses the client's MAC address.

7.2 DHCP Server Components

This section describes the software and system elements that comprise the DHCP server component, including:

- Executable files
- Configuration files
- Command files
- Logical Names
- Log files

7.2.1 Executable Files

Ten programs comprise the DHCP server component. Table 7–2 describes the programs.

Table 7–2 DHCP Executable Files

Program Name	Description
BPASCIITODBMOD.EXE	Used in rollover of old-style UCX BOOTP entries to DHCP.
BPISAMTOASCII.EXE	Used in rollover of old-style UCX BOOTP entries to DHCP.
DBDUMP.EXE	Dumps lease database in single line ASCII format. See Section 7.8.1.
DBMODIFY.EXE	Modifies lease database. See Section 7.8.2.
DBREGISTER.EXE	Registers known MAC addresses. See Section 7.8.3.
DBSHOW.EXE	Displays specified binary database. See Section 7.8.1.
GUI.EXE	DHCP GUI. Used to manage DHCP server.
SERVER.EXE	DHCP server.
SHOWDBS.EXE	Displays lease database in easy to read format. See Section 7.8.1.
SIGNAL.EXE	Implements UNIX style `kill` to allow sending signals to DHCP server. See Section 7.4.3.

7.2.2 Configuration Files

DHCP uses the configuration files described in Table 7–3 to control the behavior of the DHCP server and its service to DHCP clients.

Table 7–3 DHCP Configuration Files

File Name	Description
SERVER.PCY	Describes the behavior of the server. For example, the policy file tells you whether BOOTP clients should be supported, the ping timeout value, and so on. You may need to make modifications to this file to change the default settings. Some of the defaults include support for BOOTP clients and assigning names by IP addresses.
DHCPCAP.	Defines the client configuration parameters. This file is similar to the standard BOOTPTAB file used by most BOOTP servers. Each entry in the file can describe a single host, all the hosts within a subnet, or a group of hosts.
NETS.	Defines the pool of IP addresses available to the DHCP server to assign to clients. Used for dynamic address assignments.
NETMASKS.	Defines network masks if the network is subnetted. If you use subnetting on your network, you must enter the subnet mask into the NETMASKS. file for your network to operate correctly. This is an optional file. If your network uses standard class A, B, or C network addressing, you do not need to enter a mask into this file.
NAMEPOOL.	Defines the names available for assignment to DHCP clients. The server uses the names only as a last resort (for example, when the client did not suggest a name and there is no name associated with the IP address offered to the client).
.DDNSKEYS	Defines the domains that are to be sent DNS/BIND dynamic updates. The name of this file consists only of a file type of DDNSKEYS.

The DHCP configuration files (except for log files) are located in SYS$SYSDEVICE:[TCPIP$DHCP] or in the directory pointed to by the logical name TCPIP$DHCP_CONFIG. Log files are always located in the SYS$SYSDEVICE:[TCPIP$DHCP] directory.

Template copies of the DHCP configuration files are located in text library file SYS$LIBRARY:TCPIP$TEMPLATES.TLB. The template copies provide instructions on how to edit the text files manually.

7.2.2.1 Server Policy

The SERVER.PCY file configures the behavior of the server. This policy file describes various aspects of the server; for example, what sort of name service to use, whether BOOTP should be supported, and the ping timeout value.

Use new lines to separate entries in the SERVER.PCY file from one another. The server ignores blank lines and comments (lines beginning with the pound (#) symbol). Each new policy option must begin and end on a separate line. A keyword introduces a policy option. A policy option can be Boolean or can take a value separated from the keyword by a space (but not by a new line).

If the SERVER.PCY file contains more than one specification for an option, only the value associated with the last specification takes effect; the server disregards earlier specifications.

Example 7–1 shows the contents of the SERVER.PCY file.

Example 7–1 Sample SERVER.PCY File

```
$ TYPE PINE$DKB0:[DHCP_CONFIG]SERVER.PCY
# server.pcy: server side policy file.
#
# $Id: server.pcy,v 1.25 1997/02/24 06:22:45 robs Exp $
#
# DESCRIPTION
# This is a template server.pcy file. A particular site may need to make
# modifications to this, especially to the name service and name allocation
# policies in force
#

# Default time-to-live for an address lease if not specified on a
# per host, per subnet or per class basis.

default_ttl 86400

# Time to live on provisional list

provisional_ttl 60

# Size of the internal array specifying the number of address
# blocks held on the free list. This number should not be too
# high, or the server will "forget" about all previous allocations
# of expired leases very quickly. It should not be too low or
# performance will suffer.

free_list_size 8

# Define type of name service. The name service is one of
# { dns, local, nis, nis+}.
# local means use text files on the local system (i.e. /etc/hosts).

# On OpenVMS leave this option as "dns".
```

(continued on next page)

Example 7–1 (Cont.) Sample SERVER.PCY File

```
name_service dns

# Specify whether the name service is dynamically updateable.
# NIS and NIS+ are dynamically updateable, but the system
# administrator may choose to disable this capability. In
# both cases the server must be in the same domain as the
# name server, and the JOIN server's key must be in the
# public database. NIS also requires the creation of
# a pseudo map, "join", and the installation of the file
# "updaters" in /var/yp on the name server. See manual
# for further details. This option can be enabled for DNS.
# The default is not to permit dynamic updating.

name_service_updateable

# Name policy
# The name may be choosen according to three possible policies:
#   assign_name_by_hwaddr:
#       A particular client (identified # by its hardware address)
#       always has the same name wherever possible. This option
#       may only be choosen if the name service is updateable.
#   assign_name_by_ipaddr:
#       The client gets a name from the IP address which was
#       assigned to it, as found in the name service. This
#       option is incompatible with assign_name_by_hwaddr.
#   accept_client_name:
#       This toggle is valid only when the policy is
#       assign_name_by_hwaddr. When "on" the server will use
#       the name suggested by the client and bind it to the
#       IP address delivered by the DHCP protocol. This is
#       true even if the client in question already has a name
#       in the server's DB which is not the name suggested.
#       The old name continues to be "owned" by the client
#       and may have a valid IP address bound to it.
#       When this toggle is "off" the server will return to
#       client a pre-existing name bound to the client identifier
#       or hardware address, regardless of the name the client
#       suggests to the server.
#
# If no name can be found by the application of one or more of
# these policies, the server will generate a name for the domain
# by using the name prefix in the "namepool" database.

# assign_name_by_ipaddr

#
# Note: The following two settings are most appropriate when you are using
# dynamic DNS updates. To set this up on the DHCP server side uncomment these
# lines and delete the line above with "assign_name_by_ipaddr".
assign_name_by_hwaddr
accept_client_name

# When the naming policy is assign_name_by_hwaddr the server will
# not allow a client to use a name which is "owned" by some other
# client. I.e. A name that is already bound to a different Client
# identifier or MAC address. When this toggle is on, this prohibition
# is lifted and the name will be re-assigned

#ignore_name_owner


# Bootp.
# Remove this line if the server is not to support old-style Bootp
```

(continued on next page)

Example 7–1 (Cont.) Sample SERVER.PCY File

```
support_bootp

#This boolean is only valid if Bootp clients are supported
#(support_bootp option is enabled). When present it permits
#the server to permanently assign an IP address from its
#free pool to a BOOTP client in the event that no permanent
#binding exists in dhcpcap. Normally the JOIN server can
#only service BOOTP clients for which such a binding pre-exists.

#bootp_addr_from_pool

# Timeout value for ping in milliseconds. Before the server offers an
# address it pings (using ICMP echo) it: if a reply is received the
# server assumes that it is in use and makes another choice. "ping_timeout"
# is the number of milliseconds the server will wait for a reply.

ping_timeout 500

# Instructs the server to check whether or not the dhcpcap file appears to
# have changed each and every time a client configuration is required.
# If the file has changed (as indicated by its time stamp), the server
# will read and parse it anew.

auto_reread

# Before a BOOTP client is given a hard-wired IP address the server checks
# that the client is indeed connected to the logical IP network for which
# the address is valid. If not an error is logged and no response sent.
# In order for this to work properly the netmasks file must contain the
# network numbers and masks for any non-standard IP Class A, B or C
# configuration.

#check_bootp_client_net

# Before an IP address is given to a BOOTP client the server first checks
# to see whether or not it is in use by sending an ICMP echo. If a reply
# is received an error is logged.  If the address was from the dynamic pool
# it will be marked un-available, and a new address selected from the pool.
# If the address was statically configured the server refuses to configure
# the client.

#ping_bootp_clients

# The server will by default ignore any packets forwarded to it via a relay
# agent whose giaddr field shows it to be directly connected to the server -
# the server will, presumably, hear the clients broadcast directly. This
# option forces the server to reply regardless.

#reply_to_relay_on_local_net

# The server will not send a complete configuration to a DHCP client unless
# this toggle is set. Resolving a client configuration can be time consuming
# and, in a multi-server environment, the client may select another server.

send_options_in_offer

# Minimum packet size for DHCP requests.  By modifying this parameter,
# the DHCP server can be configured to work with some non-compliant
# DHCP clients that send DHCP requests smaller than the minimum required
# packet length.  By default, the minimum packet size is 300 bytes.

minimum_bootp_packet_size 300
```

(continued on next page)

Example 7–1 (Cont.) Sample SERVER.PCY File

```
# Set this true if you want to automatically delete leases when
# the client changes its net. I.e. if the server has leases for
# the client on several nets, and the client boots on a specific
# net, say X, the all the leases on all the nets except X, whether
# expired or not will be deleted.
#
# Note that some HW, notably SUN workstations, use a MAC address
# or client identifier which is the same regardless of the
# interface being configured. Therefore, two interfaces of a client
# of this tupe may appear to the server to be a single client
# which has changed network. You would probably not want to
# auto delete leases in this case.

auto_release

# Finite Bootp lease support. When this parameter is non-zero it
# instructs the server to grant FINITE leases to BOOTP clients.
# BOOTP clients don't know this, so before the server can re-use
# these leases it must ping the IP address. If a reply is heard
# the server automatically extends the lease by this time interval (secs).
# Note that the *original* lease conferred on a BOOTP client is
# determined by the dhcpcap file, which need not be the same as
# this extension. Also that this capability is only relevant to
# BOOTP clients which are dynamically addresses (bootp_addr_from_pool
# toggle on).

bp_auto_extension 0

# Set auto_sync_dbs to flush the server database to disk
# after each update. This is more reliable in the event
# of a failure, but slows the server down.

auto_sync_dbs
dns_tracks_dhcp_lease
# registered_clients_only
```

7.2.2.2 Client Configuration Parameter

The DHCPCAP. file describes the various configuration parameters for clients. This file is similar to the standard bootptab file used by most BOOTP servers. Each entry in the file can describe a single machine (per-node basis), all the machines within a subnet (per-subnet basis), or a group of machines (per-group basis).

Example 7–2 shows typical information found in the DHCPCAP. file. For information on how to modify the DHCPCAP. file, see Section 7.7.2.

Example 7–2 Sample DHCPCAP. File

```
$ TYPE PINE$DKB0:[DHCP_CONFIG]DHCPCAP.
# dhcpcap: database for dhcp server
#
# $Id: dhcpcap,v 1.29 1996/02/08 19:20:14 hyung Exp $
#
# DESCRIPTION
# This file is used by the server when running with
# the text database.
#
```

(continued on next page)

Example 7–2 (Cont.) Sample DHCPCAP. File

```
# Using the tc= capability to factor out identical data
# from several entries. Multiple tc's permit as many
# levels of indirection as desired.

# Be careful about including backslashes where they're needed.
# Strange things can happen otherwise.

# The data which follows is for example only. You should delete
# and add entries appropriate to configuration of your own
# networks.

# A global entry which everybody uses..

#.global:\
#       :yd=alpha.beta.gamma.com:\
#       :to=28800:

# Next entries for each subnet. . .

#subnet32:\
#       :tc=.global:\
#       :nw=192.1.1.32:\
#       :gw=192.1.1.33:\
#       :ba=192.1.1.63:\
#       :lt=7200:t1=3600:t2=6300:

#subnet64:\
#       :tc=.global:\
#       :nw=192.1.1.64:\
#       :gw=192.1.1.65:\
#       :ba=192.1.1.95:\
#       :lt=1200:t1=600:t2=1050:

# Individual entries...

# An old-style BOOTP client with the ip address hard-wired.
# This assumes that this BOOTP client will always be on
# subnet 192.1.1.32
#.xterm:\#
#       :ht=1:ha=0a0b0c0d0e0f:\
#       :ip=192.1.1.36:\
#       :bf=mybootfile:\
#       :sa=192.1.1.33:\
#       :tc=.global:

# A DHCP client. The lease time here (1day) overrides that specified in the
# network entries
#.xtermb:\
#       :ht=1:ha=0a0b0c0d0e1f:\
#       :lt=86400:\
#       :tc=.global:
 term_server:\
                                :ht=1:ha=08002ba22049:\
                                :ip=10.10.2.120:\
                                :bf=MNENG1:\
                                :bs=1159:\
                                :dn=compaq.com:\
                                :ds=10.10.2.14:\
                                :hd=sys$sysdevice[tcpip$bootp]:\
                                :sm=255.255.255.0:\
                                :td=tcpip$tftp_root:\
                                :sn=timber.compaq.com:
```

(continued on next page)

Example 7–2 (Cont.) Sample DHCPCAP. File

```
subnet_2:\
                                :nw=10.10.2.0:\
                                :gw=10.10.2.66:\
                                :ba=10.10.2.255:\
                                :lt=1200:t1=600:t2=1050:
subnet_4:\
                                :nw=10.10.4.0:\
                                :ba=10.10.4.255:\
                                :lt=1200:t1=600:t2=1050:
```

7.2.2.3 Network Addresses

The NETS. file describes the ranges of IP addresses available to the server for the clients. Both BOOTP and DHCP use this pool of addresses whenever dynamic IP assignment is needed.

Each entry in the file consists of three fields:

- The network number expressed as an IP address, for example, 142.132.3.0.
- The owner of this IP address range expressed as the IP address of the server host (142.132.3.1) or the host name (dhcpserver in Example 7–4). If a DHCP cluster failover environment is configured (see Section 7.4.5), the IP address is defined as the null address 0.0.0.0 so that applicable cluster nodes can receive packets.
- A range of available addresses for dynamic allocation to hosts on the network.

  The range is expressed as a pair of IP addresses with a hyphen (-) between them, for example, 143.32.3.10-143.32.3.30. There must be no extra space separating the dash from the IP addresses. You can specify more than one range for each network; the ranges need not be contiguous.

Example 7–3 shows the contents of the NETS. file.

Example 7–3 Sample NETS. File

```
$ TYPE PINE$DKB0:[DHCP_CONFIG]NETS.
# nets: pool of addresses available for allocation by specific join servers.
#
# $Id: nets,v 1.11 1996/01/15 17:50:00 hyung Exp $
#
# DESCRIPTION
# This file instructs the server which nets and subnets it is to administer
# and the addresses which are available for dynamic allocation.
#
# Each non-comment line in this file has up to three fields:
#    Subnet IP address
#    IP address or name of host "owning" the address range.
#    The address range itself
```

(continued on next page)

Example 7–3 (Cont.) Sample NETS. File

```
# If there are fewer than three fields then the subnet and owner
# are implied by previous entries. The address range is specified
# as one or two IP addresses. If two then they must be separated
# by a dash "-", with no whitespace intervening. Multiple ranges
# may be specified for any owner. The IP addresses are checked for
# syntax, for uniqueness of ownership, and validity on the network
# specified. If the owner of a range is multi-homed, then the
# name used must be its canonical name (e.g. as echoed by hostname),
# or, if specified by address, the address must correspond to
# the canonical name as given in /etc/hosts
#
# For OpenVMS with DHCP configured on multiple cluster nodes (ie. DHCP
# cluster failover) enter 0.0.0.0 in the "owning" DHCP server field
# (field 2).
#

# Examples:
#192.1.1.32      192.1.1.34       192.1.1.35-192.1.1.43
#192.1.1.32      192.1.2.34       192.1.1.44-192.1.1.62
#192.1.1.64      192.1.2.34       192.1.1.66-192.1.1.94
#
# DHCP cluster failover example:
#192.1.1.64      0.0.0.0          192.1.1.66-192.1.1.94
#
10.10.2.0        0.0.0.0        10.10.2.100-10.10.2.110
10.10.4.0        0.0.0.0        10.10.4.100-10.10.4.110
```

The entries in the NETS. file shown in Example 7–4 describe the IP ranges for two different networks, each with its own set of IP addresses.

Example 7–4 NETS Entries with IP Ranges for Two Networks

```
143.32.3.0  143.32.3.1 143.32.3.10-143.32.3.30  143.32.3.40-143.32.3.60
            143.32.3.75-143.32.3.100              1

143.32.5.0  dhcpserver  143.32.5.10-143.32.5.200  2
```

In this example:

1 This entry comprises two lines and describes three noncontiguous ranges of IP addresses for the network 143.32.3.0.

2 This entry describes a single range of addresses for the network 143.32.5.0. Notice the use of an IP address in the first entry (143.32.3.1) and the use of a host name (dhcpserver) in the second entry to describe the owner of the IP address ranges.

7.2.2.4 Netmask Masks

If your network is subnetted in a format that is not consistent with the standard class A, B, or C netmask address, you must include the network addresses and netmasks in the NETMASKS. file during the initial DHCP server configuration. Make sure you edit the NETMASKS. file and include an entry for each network. Each entry in the file must include two fields: the network address and the netmask address. Example 7–5 show a sample NETMASKS. file.

Example 7–5 Sample NETMASKS. File

```
$ TYPE PINE$DKB0:[DHCP_CONFIG]NETMASKS.
# Network masks. This file is only needed on those platforms
# which don't provide a netmasks database, either as a text
# file or as a map (NIS, NIS+, .. whatever).
#
# This file should contain an entry for each network for which
# the netmasks is other than the standard A,B or C mask. Each
# entry has two fields: the network and the mask. The network
# must be written with trailing zeros: e.g For net 192.1.1
# you do not enter
#
#     192.1.1
#
# but
#
#     192.1.1.0
#
#
# This file also supports variable subnetting: i.e. if each
# subnetted net can in turn be subnetted with a variable
# mask then the subnets can also appear on the LHS. Thus
#
#     192.1.1.0      255.255.255.224
#     192.1.1.96     255.255.255.240
#
# Network       netmask
10.10.2.0       255.255.255.0
10.10.4.0       255.255.255.0
```

7.2.2.5 NamePool

The NAMEPOOL. file specifies a collection of names available for dynamic assignment to DHCP clients. The server uses the names in this file only when the name is not provided another way. For example, the server might use this file when the client did not suggest a name and when there is no name associated with the IP address being offered to the client.

In addition to this pool of names, there is also a name prefix. Once the name pool is exhausted, the server generates names from the prefix by appending the number 1, 2, or 3, along with a trailing "d". After a name has been dynamically bound to a host, the server never uses the name again, even if that host subsequently acquires a new name.

Each entry in the file consists of four fields:

- The domain to which the names apply.
- The owner of these names, expressed as either the IP address of the server host (142.132.3.1) or the host name (dhcpserver).
- An optional name prefix, used for generating names after the name pool is exhausted.
- A list of names in the pool.

Example 7–6 shows the contents of a typical NAMEPOOL. file.

Example 7–6 Sample NAMEPOOL. File

```
$ TYPE PINE$DKB0:[DHCP_CONFIG]NAMEPOOL.
# namepool: pool of names available for dynamic allocation.
#
# $Id: namepool,v 1.7 1996/01/15 17:53:11 hyung Exp $
#
# DESCRIPTION
# This file contains names to be allocated to new machines coming onto the
# network. Each group of names is introduced by a single line containing
# two or three fields: the # domain name to which the names apply, the
# machine (name of address) authorized to dispense them, and (optionally)
# a prefix which will be used to generate names automatically within that
# domain. White space is used to separate these fields; there must be no
# leading whitespace on these lines.
#
# Following this are the names. These may be written one or many
# to a line, but each line must begin with a blank or tab.
#
# The character '#' introduces comments. The text following '#'
# to the end of line will be ignored by the parsing program.
# Blank lines and lines beginning with '#' are ignored.
#
# In summary format is:
#       domain_name  server  generic_name
#       TAB  hostname...
#
# Example:
# alpha.beta.gamma.com  192.1.1.65  coastal-areas
#       north-utsire south-utsire viking forties cromarty forth tyne dogger

compaq.com      timber  timber
        dhcp1   dhcp2   dhcp3   dhcp4
        dhcp5   dhcp6   dhcp7   dhcp8
        dhcp9   dhcp10
```

Example 7–7 shows a NAMEPOOL. file containing a name prefix.

Example 7–7 NAMEPOOL Entries Showing the Use of a Name Prefix

```
acme.com        142.132.3.1   pc      alpha bravo charlie delta echo
engr.acme.com   dhcpserver    EngrPC  victor whiskey xray yankee zulu
```

In this example:

- The first entry describes five names available to the acme.com domain with a name prefix of pc.
- The second entry describes five different names for the engr.acme.com domain with a name prefix of EngrPC. Notice the use of an IP address in the first entry (143.32.3.1) and the use of a host name (dhcpserver) in the second entry to describe the owner of the IP address ranges.

7.2.2.6 .DDNSKEYS

The .DDNSKEYS file describes each DNS domain and the DNS name server that is to receive Host/IP address update information when DHCP distributes an address to a DHCP client in the domain. The information in this file consists of the domain to be updated and the IP address of the DNS server to which DHCP sends the updates. A third field for secure dynamic updates is reserved for future use. TCP/IP Services does not support secure dynamic updates.

This file is required for DHCP to perform DNS dynamic updates.

The following example shows the contents of a typical .DDNSKEYS file:

```
$ TYPE PINE$DKB0:[DHCP_CONFIG].DDNSKEYS
compaq.com           10.10.2.14
10.10.in-addr.arpa   10.10.2.14
```

7.2.3 Command Files

Table 7–4 describes the command files used by the DHCP server.

Table 7–4 DHCP Server Command Files

Command File Name	Description
TCPIP$DHCP_SETUPCOMMANDS.COM	Defines symbols to invoke DHCP utilities. It is located in the SYS$MANAGER: directory.
TCPIP$DHCP_STARTUP.COM	DCL commands to start the DHCP server.
TCPIP$DHCP_CLUSTER_STARTUP.COM	DCL commands to start the DHCP server in a cluster failover configuration.
TCPIP$DHCP_SHUTDOWN.COM	DCL commands to stop the DHCP server.
TCPIP$DHCP_CLUSTER_SHUTDOWN.COM	DCL commands to stop DHCP server in a cluster failover configuration.
TCPIP$DHCP_RUN.COM	Command procedure for starting DHCP server during the startup of DHCP server.
TCPIP$DHCP_SYSTARTUP.COM	Site-specific definitions and parameters to be invoked when DHCP starts.
TCPIP$DHCP_SYSHUTDOWN.COM	Site-specific definitions and parameters to be invoked when DHCP is shut down.

7.2.4 Logical Names

By establishing logical names, you can modify the following server characteristics:

- The directory in which the DHCP configuration files and databases are placed during TCPIP$CONFIG
- Error logging and diagnostics

Table 7–5 lists the DHCP logical names and describes their function.

Table 7–5 DHCP Server Logical Names

Logical Name	Description
TCPIP$DHCP_CONFIG *directory*	If defined, places the following DHCP files (during TCPIP$CONFIG) in the directory you specify: • DHCP configuration files in ASCII format (for example, SERVER.PCY) • DHCP database files in binary format (for example, DBA.BTR) • Binary database lock files (for example, RWLOCKDBA) • Temporary files created by TCPIP$CONFIG during the BOOTP-to-DHCP rollover • The server's process identification file (JOIN.PID) Setting this logical name is useful when you want to move the file location off the system disk or when you want to set up a DHCP cluster failover environment (see Section 7.4.5). The logical name must be defined before running TCPIP$CONFIG. If not defined, the preceding DHCP-related files are placed in SYS$SYSDEVICE:[TCPIP$DHCP] during the TCPIP$CONFIG procedure.
TCPIP$DHCP_DEBUG *value*	Logs full diagnostics. Valid numeric values are 1 to 6. If you define this logical, the value of TCPIP$DHCP_LOG_LEVEL is ignored.
TCPIP$DHCP_LOG *name*	Defines the name of the DHCP server log file. The default is TCPIP$DHCP_RUN.LOG. If defined, each time the auxiliary server starts a DHCP server process, two log files are created: the one you define with TCPIP$DHCP_LOG *name* and the default TCPIP$DHCP_RUN.LOG.
TCPIP$DHCP_LOG_LEVEL *value*	Writes the specified level of diagnostic information to the log file. Ignored if TCPIP$DHCP_DEBUG is defined. Valid numeric values are: 0 No logging (default). 1 Log warning messages. 2 Log all messages.

You define system wide TCPIP$DHCP logical names in the SYS$STARTUP:TCPIP$DHCP_SYSTARTUP.COM file. After making changes to the file, enter the following commands:

```
$ @SYS$STARTUP:TCPIP$DHCP_SHUTDOWN.COM
$ @SYS$STARTUP:TCPIP$DHCP_STARTUP.COM
```

Alternatively, you can follow these steps:

1. Manually define the system logical names.
2. Use DHCPSIGHUP to signal the DHCP server.

7.2.5 Log Files

The DHCP server creates a log file named TCPIP$DHCP_RUN.LOG in the directory SYS$SYSDEVICE:[TCPIP$DHCP].

7.3 DHCP Server Startup and Shutdown

The DHCP server can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$DHCP_STARTUP.COM allows you to start up the DHCP service.
- SYS$STARTUP:TCPIP$DHCP_SHUTDOWN.COM allows you to shut down the DHCP service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$DHCP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when DHCP is started.
- SYS$STARTUP:TCPIP$DHCP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when DHCP is shut down.

7.3.1 Stopping the DHCP Server Process

If you specified automatic startup during the TCP/IP Services configuration procedure (TCPIP$CONFIG), the DHCP server process starts automatically when the DHCP service is started (TCPIP$DHCP_STARTUP.COM).

If you want to stop the DHCP server process, enter the following utility command as defined in SYS$MANAGER:TCPIP$DHCP_SETUPCOMMANDS.COM:

```
$ DHCPSIGTERM
```

Be aware that a new DHCP server process starts automatically as soon as the old process exits unless you disable the DHCP service before entering a `DHCPSIGTERM` command. As an alternative method, you can shut down DHCP by executing the following command:

```
$ @SYS$STARTUP:TCPIP$DHCP_SHUTDOWN
```

Because the DHCP server has several binary databases open (updates to which might not have been flushed to the disk), do not stop a running DHCP process using the DCL command STOP/ID=*entry_number*. Instead, stop the DHCP process by entering the DHCPSIGTERM command.

7.4 Configuring the DHCP Server

To configure the DHCP server, perform the following tasks:

Task	Described in...
Enable DHCP on your system and set up DHCP files and databases.	Section 7.4.1
Set up DNS/BIND.	Section 7.4.2
Set up the cluster failover environment.	Section 7.4.5
Stop the DHCP process.	Section 7.3.1
Shut down and start up the DHCP process.	Section 7.3
Configure client information (use the DHCP GUI or make changes manually).	Section 7.5 or Section 7.7, respectively
Set up the NETMASKS. file, if appropriate.	Section 7.2.2.4
Define IP addressing.	Section 7.6

7.4.1 Enabling the DHCP Server

To enable DHCP initially, run the TCPIP$CONFIG procedure by entering the following command and then choose DHCP from the Server Components menu:

```
$ SYS$STARTUP:@TCPIP$CONFIG
```

The configuration procedure asks if you want to convert existing BOOTP entries to DHCP database:

```
Do you want to rollover old-style BOOTP entries into the DHCP
database? [Y]
```

- If you answer Yes, the TCPIP$DHCP_BOOTPTODHCP.COM procedure tries to locate the existing BOOTP database. Once it locates a file, the configuration procedure asks you to confirm its selection or make a new selection:

  ```
  Name of file to use for old-style BOOTP:  SYS$SYSTEM:TCPIP$BOOTP.DAT
  Press return or enter new file name:
  ```

  The configuration procedure does the following to the file:

  - Converts any existing BOOTP information to the appropriate DHCP format in the new DHCPCAP. configuration file.
  - Sets up the DHCP server to support BOOTP clients.
  - Sets up permanent leases for existing BOOTP clients.

  During TCPIP$CONFIG, all DHCP-related files are placed in the SYS$SYSDEVICE:[TCPIP$DHCP] directory unless you define the system logical name TCPIP$DHCP_CONFIG (see Table 7–5).

- If you answer No, the new DHCP configuration file DHCPCAP. remains empty, and your BOOTP clients will not be served.

  TCPIP$CONFIG invokes the command procedure SYS$MANAGER:TCPIP$DHCP_SETUPCOMMANDS.COM, which defines the GUI Server Management Console and DHCP utilities as OpenVMS foreign commands.

Important

Compaq recommends calling the TCPIP$DHCP_SETUPCOMMANDS.COM procedure as part of the login process for all users who are authorized to manage the DHCP server.

7.4.2 Configuring DHCP and DNS/BIND to Assign Host Names

DHCP uses the following methods to assign a host name:

- By hardware address

  When you specify this method, DHCP uses the host name suggested by a client when the client sends out its initial boot request.

  This method requires that both the DHCP and BIND servers are capable of and configured for performing dynamic DNS updates.

  To configure host name assignment using this method, see Section 7.4.2.1.

- By IP address

  If you specify this method to assign host names, DHCP performs a BIND IP address lookup to obtain a host name associated with the IP address. If the lookup is successful, DHCP uses the host name returned by BIND. If the lookup fails, DHCP creates a name from the NAMEPOOL. file.

  This method requires that you manually assign an IP address to each host and add A and PTR records to your DNS/BIND database.

  To configure host name assignment using this method, see Section 7.4.2.2.

7.4.2.1 Dynamically Assigning Host Names

To configure DHCP to assign a host name dynamically, perform the following steps:

1. Change the SERVER.PCY file (either manually or with the DHCP GUI) to include the following statements:

   ```
   name_service_updateable
   assign_name_by_hwaddr
   accept_client_name
   dns_tracks_dhcp_lease
   ```

2. Configure a DNS/BIND server to accept dynamic updates from your DHCP server. If you are running the DHCP server on multiple nodes, configure the DNS/BIND server to accept dynamic updates from each of the nodes. Refer to Section 5.3.6 for a discussion on how to configure DNS/BIND to accept dynamic updates from DHCP.

3. Edit the DHCPCAP. file to add a domain name for all subnet entries for which DHCP will perform dynamic DNS updates. To use the DHCP GUI to add dynamic DNS updates for a domain, do the following:

   a. Start the DHCP GUI as described in Section 7.5

   b. Select the Subnets tab.

   c. Select DHCP parameters from the drop down list

   d. Add the domain name to the DNS Domain Name parameter.

4. Create a .DDNSKEYS file with an entries for the DNS/BIND server that is to receive dynamic updates. You will most likely want to create an entry for A and PTR records by defining a forward and reverse translation entry.
5. Create a NAMEPOOL. file to supply a pool of names to use for nodes on the particular network. DHCP uses this pool of names to generate a host name only when other methods are unsuccessful.

7.4.2.2 Statically Assigning Host Names

To configure DHCP to use host names defined in a DNS/BIND server database, perform the following steps:

- Change the SERVER.PCY file (either manually or with the DHCP GUI) to include the following statement:

  ```
  assign_name_by_ipaddr
  ```

  Exclude the following statements:

  ```
  accept_client_name
  dns_tracks_dhcp_lease
  name_service_updateable
  ```

  See Section 7.2.2.1.

- Edit the DNS/BIND forward translation (*domain_name*.DB) file to include an A record for each IP address in the range of IP addresses that your DHCP server may allocate. It is common practice to assign IP addresses and names systematically. For example, if IP address 10.10.2.100 obtains the name dhcp1.xyz.com, then IP address 10.10.2.101 obtains the name dhcp2.xyz.com (see Section 5.4.4.3).
- Edit the DNS/BIND reverse translation (*address*.DB) file to include a PTR record for each host name (see Section 5.4.4.4).

7.4.3 Signaling the DHCP Server

One of the DHCP utilities that is defined in TCPIP$DHCP_SETUPCOMMANDS.COM is the TCPIP$DHCP_SIGNAL utility, which provides interprocess signaling in a manner similar to the UNIX kill signal delivery utility. PRMMBX and SYSNAM privileges are required to run TCPIP$DHCP_SIGNAL.EXE.

The following table shows the commands available with the TCPIP$DHCP_SIGNAL utility:

Command	Description
DHCPSIGHUP	Causes the ASCII configuration files to be read again, flushes the binary databases and then translates the TCPIP$DHCP_DEBUG and TCPIP$DHCP_LOG_LEVEL logical names.
DHCPSIGTERM	Causes an orderly shutdown of DHCP.
DHCPSIGUSR1	Causes a dump of the ASCII configuration files, then flushes the binary databases.

7.4.4 Returning to the BOOTP-Only Configuration

You can return to a BOOTP-only configuration at any time. Further, you can use the previous TCPIP$BOOTP.DAT database file and the client entries it contains. If you deleted the TCPIP$BOOTP.DAT file, you can create a new one and populate it with entries (see Section 9.5).

To enable BOOTP after you have configured your host for DHCP, run TCPIP$CONFIG and enable the BOOTP component from the Server Components menu. Your existing DHCP files will remain for future use.

7.4.5 Setting Up a DHCP Cluster Failover Environment

You can set up an OpenVMS Cluster environment for DHCP server failover. In this environment, a standby system becomes the DHCP server if the active DHCP server process fails or is stopped, or the system on which it is running fails or shuts down.

With cluster failover, the DHCP server uses the OpenVMS lock manager during process initialization to acquire a system-level, exclusive-mode lock on a resource called TCPIP$DHCP_SERVER. The first server started on the cluster obtains the lock on TCPIP$DHCP_SERVER and becomes the active DHCP server. The other DHCP servers wait to obtain the lock and become the standby servers.

When the active DHCP server process exits for any reason, the lock on TCPIP$DHCP_SERVER is released and one of the standby processes acquires the lock and becomes the active server.

To configure the DHCP server failover environment, do the following:

1. If the DHCP server is running on one of your systems, manually disable it by entering the following command on the server system:

   ```
   $ @SYS$STARTUP:TCPIP$DHCP_SHUTDOWN.COM
   ```

2. Create a directory for the DHCP configuration and binary database files that is visible to the DHCP cluster members. Specify TCPIP$DHCP as the directory's owner. For example:

   ```
   $ CREATE/DIRECTORY/OWNER=TCPIP$DHCP WORK1$:[DHCP_CONFIG]
   ```

3. If you have already been running DHCP server and want to start with the existing data files, then do the following:

   a. Copy the DHCP data files from the DHCP directory to TCPIP$DHCP_CONFIG:*.* by entering commands similar to the following:

      ```
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]DHCPCAP. TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]DHCPTAGS. TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]NAMEPOOL. TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]NETMASKS. TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]NETS. TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]SERVER.PCY TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP]DB%.%%% TCPIP$DHCP_CONFIG:
      $ COPY SYS$SYSDEVICE:[TCPIP$DHCP].DDNSKEYS TCPIP$DHCP_CONFIG:
      ```

b. Delete the DHCP data files from the DHCP directory by renaming them to a temporary subdirectory. (You can delete the files after you are sure that the failover environment is set up correctly.) For example, enter the following commands:

```
$ CREATE/DIR SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ PURGE SYS$SYSDEVICE:[TCPIP$DHCP]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]DHCPCAP.;*  SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]DHCPTAGS.;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]NAMEPOOL.;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]NETMASKS.;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]NETS.;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]SERVER.PCY;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP]DB%.%%%;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
$ RENAME SYS$SYSDEVICE:[TCPIP$DHCP].DDNSKEYS;* SYS$SYSDEVICE:[TCPIP$DHCP.SAVE]
```

4. On each cluster node that is to serve as a potential DHCP server, set up the TCPIP$DHCP_CONFIG logical name as follows:

   a. Define TCPIP$DHCP_CONFIG as a systemwide logical name. For example:

      ```
      $ DEFINE/SYSTEM TCPIP$DHCP_CONFIG WORK1$:[DHCP_CONFIG]
      ```

   b. Before you run the TCPIP$STARTUP.COM procedure, add the TCPIP$DHCP_CONFIG logical name definition to the TCPIP$DHCP_SYSTARTUP.COM file.

5. On each cluster node that you want to be a DHCP server, run TCPIP$CONFIG to enable the DHCP service (see Section 7.4.1).

   The TCPIP$CONFIG procedure creates the TCPIP$DHCP account and stores initial copies of the DHCP configuration data files in the directory pointed to by the logical name TCPIP$DHCP_CONFIG. If you choose to roll over your BOOTP database to DHCP, TCPIP$CONFIG creates your initial DHCP binary database files in the TCPIP$DHCP_CONFIG directory.

6. Make sure that the auto_sync_dbs parameter is set in the SERVER.PCY file.

   This parameter causes the DHCP server databases to be flushed after each update. You can set the parameter by editing the SERVER.PCY file or by setting the auto_sync_dbs parameter to True on the Server/Security tab in the DHCP GUI.

7. Ensure that the files in TCPIP$DHCP_CONFIG: and the directory itself are owned by TCPIP$DHCP and have owner-only protection (O:RWED). For example:

   ```
   $ DIRECTORY/SECURITY WORK1$:[DHCP_CONFIG]
   $ DIRECTORY/SECURITY WORK1$:[000000]DHCP_CONFIG.DIR
   ```

8. Edit the NETS. file and set the ownership of any existing IP address range to 0.0.0.0.

With the DHCP cluster failover configured, you need to indicate that an address range is owned by other hosts. Therefore, you specify the null IP address of 0.0.0.0 in the second field of the NETS. file in each IP address range to be shared among the DHCP servers. For example, the following entry in the NETS. file is owned by IP address 17.18.208.100:

```
17.18.0.0       17.18.208.100      17.18.208.10-17.18.208.50
```

You would change the entry to the following:

```
17.18.0.0       0.0.0.0       17.18.208.10-17.18.208.50
```

If you prefer to use the DHCP GUI to configure the null address, choose the IP Ranges parameter on the Server/Security tab and set the parameter to True.

9. Shut down DHCP on each cluster member where DHCP is running by using the TCPIP$DHCP_CLUSTER_SHUTDOWN command procedure. When the command procedure is finished, restart DHCP on the cluster by using TCPIP$DHCP_CLUSTER_STARTUP.

7.4.6 Methods to Configure DHCP Parameters

TCP/IP Services provides three methods for configuring server and client parameters:

- An easy-to-use DHCP graphical user interface (GUI) to do the following:
  - Configure dynamic and static IP addressing for all clients. See Section 7.6.
  - Configure the client information appropriate for your client base. See Section 7.5.
  - Set DHCP parameters to customize the DHCP server. See Section 9.4.3.
- Manually editing the DHCP configuration files and then signaling the DHCP server to read the files. See Section 7.7.
- Using the DHCPDBMOD utility. See Section 7.8.2.

7.5 Using DHCP GUI to Configure DHCP

You can modify the default DHCP server settings and define additional characteristics by performing the following tasks:

Task	Described in...
Define server and security parameters.	Section 7.5.2
Define subnet parameters.	Section 7.5.3.1
Define node parameters.	Section 7.5.3.2
Define group parameters.	Section 7.5.3.3

7.5.1 General Information

To use the DHCP GUI to configure DHCP:

- You need the following system privileges:

  BYPASS
  SYSNAM
  PRMMBX

- If you have not already done so, execute the TCPIP$DHCP_SETUPCOMMANDS.COM command procedure to establish DHCP foreign commands .
- Invoke the GUI by entering the following utility program command:

  ```
  $ DHCPGUI
  ```

The system displays the configuration window with four tabs across the top of the window. The tabs allow you to configure the following sets of DHCP parameters:

Tab	Function
Server/Security	Defines the server configuration (see Section 7.5.2). You can set your IP address ranges, general server parameters, or view currently leased IP addresses and their lease time.
Subnets	Assigns client configurations for entire subnets.
Nodes	Adds and customizes specific machines on your network, usually for BOOTP clients.
Groups	Defines a group of settings for predefined collections of machines.

Choose a tab for the category of parameters you want to configure. The window for each tab has three columns:

- The left column lists the items that are configured for that category. The list always contains a [New Record] item to configure another machine. Choose an item from this list to enter or view its parameters.
- The middle column lists the available parameters for the selected item along with the current specification or setting. Choose a parameter to enter or change the specification or setting.
- The right column has fields for entering data. To add or change a parameter setting, select a parameter and enter the value for the parameter in the field to the right. You can enter values as:
  - Descriptive text, such as an IP address
  - Time in hours, minutes, and seconds
  - True or false statements

  When there is more than one value field, press Tab to move to the next field. To delete information in a field, select the text, then click Delete.

7.5.1.1 Saving Information in a Record

If you add or revise information in a field, you need to save the information using one of the following methods:

1. Choose Update from the File menu.
2. Choose Exit from the File menu, then choose Save and Exit. This updates the database when you exit the program.

7.5.1.2 Adding New Records

For some subjects, you can add more than one record. To add a new record:

1. Choose [New Record] from the list on the left side of the window.
2. Enter the information for the new record.
3. Choose and enter parameter information as appropriate.

When only one record is possible, [New Record] disappears after you configure the first server.

7.5.2 Configuring Server and Security Parameters

Use the Server/Security tab to perform the following tasks:

- Configure Server/Security parameters.
- Configure IP ranges.
- Set up the host name lists.
- Use the Active IP Snapshot window.
- Preload MAC addresses.

7.5.2.1 Server/Security Parameters

To configure the server parameters using the Server/Security tab of the GUI, follow these steps:

1. Click the Server/Security tab.
2. Choose a parameter class from the drop-down list.
3. Choose the parameter you want to change.

You can change any or all of the Server/Security parameters described in this section.

Accept Client Name

Specifies whether the server assigns names to client machines according to a policy that is established on the server by the system manager.

Even when this capability is enabled, the server ignores the client-suggested name if it is already in use by another client in the same domain.

If the server is unable to find a name for the client by applying this policy, it will take one of the following actions depending upon the specified value. The valid values are:

False: Assign a name from the NAMEPOOL. file if an IP address lookup does not return an associated name. Default.

True: Use the name the client suggests for itself, if specified.

Assign Name by Hardware Addr

Specifies whether you can assign host names by the hardware address. If you choose True, the client computer always has the same name, even if its IP address changes; however, to do so, the client must remain in the same domain.

This option is appropriate for sites supporting dynamic updating of the name service. When you select this policy, the server maintains a binding of the client's unique identifier to the name the client first acquires.

If the name service does not dynamically update, the new name-IP address mapping implied by this policy is not available to other clients until you bring the name service up to date by another mechanism. This means dumping data from the database and using it to update the name service manually. The following are valid values:

False: Disable assignment of host names by hardware addresses. Default.

True: Enable assignment of host names by hardware addresses. Use the naming method defined in the NAMEPOOL. file.

Assign Name by IP Addr

Specifies whether you can assign host names by an IP address. If you choose True, the client receives its name from the name service as a result of a `gethostbyaddr` routine call. Also, when a client computer moves, it can receive a new name from the name service. The following are valid values:

False: Host names cannot be assigned by IP addresses. The DHCP server does not issue a `gethostbyaddr` routine call. Instead, the session uses the naming method defined in the NAMEPOOL. file.

True: Host names can be assigned by IP addresses. Default.

Auto Release Old Lease

Set this to True if you want to automatically delete leases when the client changes its network. For example, if the client:

- Receives an address on subnet A
- Then moves to subnet B

The server releases the leased IP address on subnet A even though the leased IP address on subnet A is still valid.

The default setting is False.

Note

Some hardware configurations use a MAC address or client identifier that is the same regardless of which interface you are configuring. To the DHCP server, two interfaces of a client of this type can appear to be a single client that has changed networks. Do not autorelease these leases.

Auto Reread Config File

Instructs the server to see if the DHCPCAP. file has changed, indicated by the timestamp. This occurs each time a client requires a configuration. If the file changes, the server rereads and reparses the DHCPCAP. file.

The default is True.

Auto Synchronize Database

Choose True to flush the server database to disk after each update. This makes the server more reliable if there is a failure such as a system crash or unintentional power shutdown. Setting this parameter to True can slow down the server.

The default is False.

BOOTP Addr From Pool

Specifies whether the DHCP server does not require a preestablished binding for BOOTP clients. When none exists, the server allocates an address from the pool to the client. Because BOOTP does not understand the concept of lease times, all such allocations are permanent regardless of the lease times specified elsewhere in the database.

When you disable BOOTP Addr From Pool, the Server only supports BOOTP clients whose IP address is configured into the database. This means the binding of the IP address to the client must be preestablished. The address must be consistent with the network to which the client is attached. See Section 7.6 for information on how to preestablish a binding between a MAC address and an IP address. The following are valid values:

False: Do not pick an address from a pool. Requires a preestablished binding. Default.

True: Pick an address from a pool. Does not require a preestablished binding.

BOOTP Client Lease Extension

TCP/IP Services does not currently support this parameter.

When you set this parameter to a value above zero, the server grants Finite leases to BOOTP clients. BOOTP clients do not know this, so before the server can reuse these leases, it must ping the IP address. If the server hears a reply, it extends the lease by the time interval (in seconds) specified by this parameter.

The default value is 0 seconds.

BOOTP Compatibility

DHCP can serve BOOTP clients as well as DHCP clients. The following are valid values:

False: The server should act as a DHCP server only.

True: The server should also act as a BOOTP server. Default.

Bootfile not sent as option

Because the DHCP clients normally do not require bootfile names, the space reserved for this purpose (the “file” field) in reply packets is used by JOIN as an extension of the DHCP options field. This arrangement permits the client to receive more configuration information than would otherwise be possible in a standard-sized DHCP packet.

Enabling this parameter sends the bootfile name in the “file” field, instead of as a DHCP option. Certain network computers (NCs) expect to find the bootfile information in the “file” field and will not successfully load their OS images unless this parameter is set to True. Note: BOOTP clients always receive a bootfile name in the “file” field, regardless of the state of this option.

Canonical Name

Overrides the value normally returned by a `gethostname` routine call (default). Primarily used for multihomed hosts with a canonical name corresponding to an interface that is not recognized by DHCP (for example, ATM interfaces) and for high-availability servers that have per-service IP addresses that differ from a physical IP host address. The following are valid values:

False: Use the host name returned by a `gethostname` routine call. Default.

True: Use the specified canonical host name.

Check BOOTP Client Net

Before a BOOTP client is given a hard-wired IP address, the server makes sure that the client is connected to the logical IP network for which the address is valid. If the client is not connected, the server logs an error and does not send a response to the client.

For this to work properly, the NETMASKS. file must contain the network numbers and masks for any nonstandard IP Class A, B, or C configuration. The following are valid values:

False: Do not check the IP network of the address. Default.

True: Check the IP network of the address.

DNS expiration tracks DHCP lease

This parameter implies that SIG resource records for DNS are updated with expiration times that match the DHCP client's lease time. If a client sends a "DHCP release", the lease is prematurely expired and the SIG record is marked as expired. In order to reduce the amount of traffic between DHCP and DNS, the default value is False.

This policy affects only installations that use the Dynamic DNS option.

Default Lease Time

Specifies the value used on all leases for clients that have no other value explicitly configured. Enter the lease time of the IP address granted to a client.

The default lease time is one day.

Expand BOOTP Packet

Expands the BOOTP reply packet to 548 bytes. Applies to BOOTP clients only. The following are valid values:

False: All replies to BOOTP clients are 300 octets or a size equal to the size of the packet received, whichever is larger. Default.

True: All replies to BOOTP clients are expanded to 548 bytes.

Force Broadcast Reply

The DHCP server generally sends unicast reply packets in response to client packets. This toggle tells the server to send broadcast reply packets instead of unicast reply packets. The following are valid values:

False: Forces the DHCP server to use unicast reply packets. Default.

True: Forces DHCP server to broadcast reply packets to the client, even when the server could use unicast replies.

Free List Size

Specifies the size of the internal array specifying the number of address blocks held on the free list. If this number is too high, the server will lose previous allocations of expired leases quickly. If this number is too low, performance can suffer.

The default setting is 8.

Ignore Hardware Type

This toggle tells the server to use the clients' hardware address as its identifier (for those clients that do not use DHCP client identifiers), but to ignore the hardware type field. In the DHCP DB the identifier is stored with a type field of zero (which is also the type for those clients which are using client identifiers).

Set this option to True only to work around problems introduced by clients that broadcast multiple DHCP requests with conflicting hardware types (for example, HP Jet Direct). The default value is False.

Ignore Name Owner

This parameter applies only if both "Assign Name by Hardware Address" and "Accept Client Name" are True. In such a case, a previously established name-hardware address binding with the same name will be overwritten with the MAC address of the requesting client in DHCP's internal name database.

Listen on PPP Interfaces

Not currently supported.

If True, the server will respond to DHCP requests on Point-to-Point Protocol (PPP) interfaces of the host. By default, DHCP ignores these interfaces.

Min BOOTP Packet Size

Specifies the minimum packet size for DHCP requests. Change this value to allow the Server to work with some noncompliant DHCP clients that send DHCP requests smaller than the minimum required packet length.

The default minimum packet size is 300 bytes.

Name Service

Specifies the implementation of the underlying name service. Name service authenticates, routes, addresses, and performs naming-related functions for other computers on the network.

DNS is the only name service available with TCP/IP Services.

Name Service Updatable

Choose True to have TCP/IP Services automatically update the name service with the assigned IP addresses and host names.

Ping BOOTP Clients

Before the DHCP server assigns an IP address to a BOOTP client, the server checks to see if the address is available by using ping to send an Internet Control Message Protocol (ICMP) echo request. If the server receives a reply, it logs an error. Then:

- If the address was from the dynamic pool, the server marks it as unavailable, and selects a new address from the pool.
- If the address was statically configured, the server refuses to configure the client.

The following are valid values:

False: Do not send an ICMP echo request to a BOOTP client before assigning an IP address. Default.

True: Send an ICMP echo request to a BOOTP client before assigning an IP address.

Ping Timeout

Specifies the duration (in milliseconds) of the ping timeout. Enter the amount of time the server is to wait before concluding no other host is using the IP address. After the timeout, the ping command stops checking.

If you do not want the server to ping before giving out an IP address, set the timeout value to 0.

The default is 500 milliseconds.

Provisional Time To Live

Specifies the maximum time period that an IP address can remain on the provisionally allocated list before it can be allocated to another client. The value should be limited to a few minutes.

The default is 1 minute.

Reply to Relay On Local Net

Specifies whether the server ignores packets forwarded to it from a relay agent on the same subnet as the server.

The following are valid values:

False: Do not reply (the server should hear the client broadcast directly). Default.

True: Reply no matter where the agent is located (the value in giaddr field).

Restrict to Known MAC Addresses

Specifies whether to restrict IP addresses that are assigned to a matching MAC address. When specified, you can manually assign a MAC address. This parameter indicates whether the server should respond to clients with a MAC address that is unknown to the server.

Choose True to have the server provide DHCP information to only those hosts that have a known MAC address. To register a known MAC address client, use `Preload MAC Addresses` feature from the Server/Security tab or use the DBREG utility.

The following are valid values:

False: Do not allow manual assignment of MAC addresses. Default.

True: Allow manual assignment of MAC addresses.

Send Options in DHCP Offer

Specifies whether the server is to send a complete configuration to a DHCP client. Resolving a client configuration can be time consuming. In a multiserver environment, the client can select another server.

The following are valid values:

False: Send a minimum configuration. Default.

True: Send a complete configuration.

Support Microsoft RAS Server
Specifies support for the Microsoft Proxy Remote Access Server (RAS). The RAS server generates a BOOTP packet with a MAC address of 16 octets.

The following are valid values:

False: Ignore a BOOTP packet with a MAC address of 16 octets. Default.

True: Recognize a BOOTP packet with a MAC address of 16 octets.

Use MAC addr as client ID
Specifies whether the server is to use the client ID to uniquely identify a client. If set to True, the server uses the client's MAC address as the client ID. BOOTP also uses the MAC address to uniquely identify a client.

The following are valid values:

False: Use client ID to identify clients. Default.

True: Use MAC address to identify clients.

7.5.2.2 Configuring IP Ranges

Use the IP Ranges parameters to specify the IP addresses that are available to assign to clients.

Note

If your network contains subnets, that information must be included in the NETMASKS. file. See Section 7.2.2.4 for more information on the use of netmasks when you are using subnet addressing.

To configure the server IP ranges:

1. Click the Server/Security tab.
2. Choose IP Ranges from the drop-down list.
3. Choose [New IP Range].
4. For each IP range, enter the subnet address or name, a server address, and an IP range to be assigned to clients on the selected subnets.

IP Range Parameters

You can change any or all of the IP range parameters described in this section.

Subnet Address
Enter the subnet address or name.

DHCP Server (address)
Enter the IP address or the name of the Server. For cluster failover configurations, enter `0.0.0.0` for the IP address.

IP Ranges
The IP Address Range is a group of unique IP addresses that the server can assign to clients on a selected subnet. To assign an IP Address Range to a subnet:

1. Enter the beginning of the IP Address Range for the subnet: network, subnet, and host address.
2. Enter the end of the IP Address Range.

3. If your network has more than one subnet, enter the remaining subnet IP numbers.

Note

A subnet address can have more than one corresponding IP Address Range.

The server can configure clients on more than one subnet when the routers between the server and the client forward BOOTP packets.

7.5.2.3 Configuring Host Names

Use the Host Names Lists Parameters to configure a host name. If you have set the server configuration so that the server automatically accepts the name a client suggests for itself or you have added A and PTR records for the hosts to your DNS/BIND database, you do not need to set up host names.

Note

Follow the instructions in this section only if the Accept Client Name parameter is set to False.

To configure a host name:

1. Click the Server/Security tab.
2. Choose Host Name Lists from the drop-down list.
3. Choose [New Host name List].
4. For each host name, enter:
   - Domain name
   - DHCP server name
   - Host name Prefix (required as part of the host name)
   - Host names

7.5.2.3.1 Host Name List Parameters You can use the following host name list parameters to set up host names.

Domain Name
Specifies the domain name. Enter the domain name exactly as it was assigned by the NIC Domain Registrar, including its top-level domain extension. For example, enter `school.edu`, `company.com`, or `city.gov`.

DHCP Server
Enter the IP address or name of the DHCP server.

Hostname Prefix
Specifies a host name prefix.

The host name prefix is used when a computer requests a host name and one is not available.

Using the `mycompany.com` domain as an example, assume:

- All names in the host name list have been assigned.

- Host name prefix is magic.

Then, the DHCP server gives the host names `magic1` and `magic2` to the next two computers that request host names.

Enter a specific host name prefix.

Host names
Specifies the list of host names. Enter as many host names as needed. Different DHCP servers can own the same host names.

7.5.2.4 Active IP Snapshot

You can use the Active IP Snapshot window to view the lease database, manually add a new lease, and remove a lease.

Viewing a Lease
The left side of the Active IP Snapshot window lists each DHCP client with a lease granted by the server. To see the details:

1. Click the Server/Security tab.
2. Choose Active IP Snapshot from the drop-down list.
3. Select a record on the left side of the window.
4. Review the information on the right side of the window. It lists the information that applies to the selected record.

Adding a New Lease
Typically you only add a new lease when you intend to permanently attach a hardware address to an IP address. The IP address does not need to come from the DHCP IP addresses you have defined.

To add a new lease, use the following procedure:

1. Click the Server/Security tab.
2. Choose Active IP Snapshot from the drop-down list.
3. Choose [New Record].
4. Enter a value for each parameter.
5. Click Add.

Changes made to the database take effect immediately.

Note

Ensure that the IP address you specify does not belong to any pool of IP addresses configured in an IP range. If it does, it could be released and used by other clients (MAC address).

If you want to grant a lease for an infinite period of time, which effectively make a permanent binding between an IP address and a MAC address, set the Lease Expiration parameter to a value of -1.

Removing a Lease
To remove a lease, use the following procedure:

1. Click the Server/Security tab.

2. Choose Active IP Snapshot from the drop-down list.
3. On the left side of the window, select the record you want to remove.
4. Click Delete.

Changes to the database take effect immediately.

Refreshing the Active IP Snapshot Window

To refresh the Active IP Snapshot window so that it reflects the current status of the database, click Refresh. This parameter will refresh data on leases that are active or expired, or both.

7.5.2.5 Preload MAC Addresses

Use the Preload MAC Addresses window to restrict the assignment of IP addresses. To enable this security measure, set the Restrict to known MAC addr value to True in the Server/Security Parameters window. You can then manually assign the desired MAC addresses. The server ignores all other client DHCP requests.

Checking the Status of a MAC Address

Each configured MAC address and type is listed on the left side of the Preload MAC Addresses window. To see the details of a MAC address:

1. Click the Server/Security tab.
2. Choose Preload MAC Addresses from the drop-down list.
3. Select a record from the left side of the window.

The right side of the window lists the information applicable to the address.

Adding a New MAC Address

Initially, you may need to add large numbers of MAC addresses to the known clients database; it may be more practical to use the command line utility `jdbreg` for this purpose. You would typically use the GUI to add MAC addresses when new (trusted) clients appear on the network.

To add a new MAC address:

1. Click the Server/Security tab.
2. Choose Preload MAC Addresses from the drop-down list.
3. Choose [New Record].
4. Enter a value for each parameter.
5. Click Add.

Changes to the database take effect immediately.

Removing a MAC Address

To remove a MAC address:

1. Click the Server/Security tab.
2. Choose Preload MAC Addresses from the drop-down list.
3. Choose the MAC address you want to delete.
4. Click Delete.

Changes to the database take effect immediately.

Searching for a MAC or IP Address

To search for a MAC or IP address:

1. Click the Server/Security tab.
2. Choose Preload MAC Addresses from the drop-down list.
3. Click Find.
4. Enter the MAC or IP address you want to locate.
5. Click OK.

Refreshing the MAC Addresses Window

To refresh the MAC address window so that it reflects the current status of the database, click Refresh.

7.5.3 Configuring Parameters for Clients

DHCP allows you to configure many client parameters in addition to the client's IP address. For example you can configure the IP address of a client's bind server and its DNS domain name.

There are three ways to assign configuration parameters to DHCP clients. You can assign parameters to:

- A specific client

  To assign parameters to a specific client, use the node configuration options described in Section 7.5.3.2. This is called a node group and is identified by the client's MAC address.

- A net or subnet

  To assign parameters to all clients in a subnet, use the subnet configuration options described in Section 7.5.3.1. This is called a subnet group and is identified by the subnet in which the client is being configured. You will likely configure each client within a subnet with similar parameters, for example, DNS domain name, DNS server, default route, and so forth.

- A group of clients

  To assign parameters to a group of clients that are not in the same subnet, use the group configuration options described in Section 7.5.3.3. This is called an include group. You can declare a node or subnet group as a member of an include group to pull in the include group's parameters.

After the DHCP server finds an IP address for a client, it matches the client's MAC address against your node groups and the client's subnet against your subnet groups, pulling any parameters from matched groups into the list of parameters to be sent to the DHCP client. If a match occurs against both a subnet and a node group, and a particular parameter is assigned in both the subnet and the node group, then the value from the node group is used. When a match occurs on a subnet or node group that is a member of an include group, the DHCP server pulls in parameters from the include group also.

7.5.3.1 The Subnets Tab

A subnet is a segment of a logical network that has been divided into smaller physical networks. Use the Subnets tab to configure parameters to be passed to DHCP clients according to the subnet in which they reside.

7.5.3.1.1 Configuring a subnet You do not have to change every value for the parameters in the Subnets tab.

To configure a subnet group using the Subnets tab, use the following procedure. For a description of the subnet parameters, see Section 7.5.3.4.

1. Click the Subnets tab.
2. Choose [New Record].
3. Choose the Name parameter from the Name/ID Parameters menu.
4. Enter the name of the subnet configuration in the Value field. This name is a tag for internal use of the DHCP server only. For more information, see Section 7.5.3.4.1.
5. Choose Member of Group (optional). Enter the name of the include group that the subnet group is joining. Any client that matches this entry will pull in the parameters from the specified include group.
6. Set up key information:
   a. Choose the net or subnet IP address.

      Enter the net or subnet IP address that identifies the subnet portion of the network.

   b. Choose the vendor class (optional).

      Enter the vendor class (for example, TCPVMS or JOIN) that identifies the DHCP client vendor class to which this entry should apply. Note that you can have multiple subnet entries for the same Net or Subnet IP Address if they have different Vendor Class key values. If the entry should apply to any vendor class or you are not using vendor classes leave the Vendor Class field blank.
7. Choose from the lists of DHCP parameters on the drop-down list.

   Different lists of DHCP parameters are available on the drop down list. Choose either BASIC DHCP Parameters or DHCP Parameters.
8. As appropriate, enter information for Network, Lease, Time, BOOTP, NetBIOS, X Window, TCP, IP, and Link parameters. For more information about these parameters, refer to Section 7.5.3.4.
9. Choose Update from the File menu to update the server with the new configuration.

The new configuration takes effect immediately.

7.5.3.1.2 Removing a Subnet Record To remove a subnet record:

1. Click the Subnets tab.
2. Choose DHCP Parameters from the drop-down list.
3. Choose the subnet record you want to delete.
4. Click Delete.

Changes to the database take effect immediately.

7.5.3.2 The Nodes Tab

A node is a workstation, computer, or other device on the network. Use the Nodes tab to configure parameters to be passed to specific client nodes.

7.5.3.2.1 Configuring a node You need not change every value for the parameters in the Nodes tab. A node group can be a member of an include group although the settings for a node group override those from a subnet or include group.

To configure a node group using the Nodes tab, use the following procedure. For a description of the node parameters, see Section 7.5.3.4.

1. Click the Nodes tab.
2. Choose [New Record].
3. Enter the name of the node configuration in the Value field. This name is a tag for internal use of the DHCP server only. For more information, see Section 7.5.3.4.1.
4. Choose Member of Group (optional). Enter the name of the include group that the node group is joining. The client that matches this entry will pull in the parameters from the specified include group.
5. Set up key information:
   a. Choose Hardware Address. Enter either the hardware address or the client ID of the node.

      If you are using the hardware address (MAC address) of the node, enter it using the format *xx:xx:xx:xx:xx:xx*, for example, 00:08:C7:08:E3:63. The hardware address is assigned during manufacturing, and usually appears when you turn on or reboot your computer.

   b. Choose Hardware Type. Enter the type of network the node is connected to: Token Ring, Ether3, Pronet, ARCnet, or 0 (see Table 7–6).
6. Choose from the lists of DHCP parameters on the drop-down list.

   Different lists of DHCP parameters are available on the drop-down list. Choose either BASIC DHCP Parameters or DHCP Parameters.
7. As appropriate, enter information for Network, Lease, Time, BOOTP, NetBIOS, X Window, TCP, IP, and Link parameters. For more information about these parameters, refer to Section 7.5.3.4.
8. Choose Update from the File menu to update the server with the new configuration.

The new configuration takes effect immediately.

Table 7–6 Network Type Symbol and Number

Symbol	Number	Network Type
ethernet or ether	1	10 MB Ethernet
ethernet3 or ether3	2	3 MB experimental
ax.25	3	AX.25 Amateur Radio
protnet	4	Protnet proNET Token Ring
chaos	5	Chaos
token-ring,tr,ieee802	6	IEEE802
arcnet	7	ARCnet

7.5.3.2.2 Removing a node record To remove a node record:

1. Click the Nodes tab.
2. Choose DHCP Parameters from the drop-down list.
3. Choose the Node record you want to delete.
4. Click Delete.

Changes to the database take effect immediately.

7.5.3.3 The Groups Tab

An include group is a collection of parameters to be passed to a set of workstations or other computers on the network which can be on different subnets. Use the Groups tab to configure include groups.

7.5.3.3.1 Using group parameters You can define a group so that a set of workstations, possibly on different subnets, has the same configuration values. For example, you might want a group to include specific lease time information for your network environment and you want this lease information to be used for all of your clients. You can define an include group holding this lease information and make your subnet groups members of this include group. The alternative would be to duplicate the lease information in each individual subnet group entry which is more difficult and error prone. Include groups can be members of other include groups. This allows you to create hierarchies of available network services across many clients.

7.5.3.3.2 Defining a group To define an include group using the Groups tab, use the following procedure. For a description of the group parameters, see Section 7.5.3.4.

1. Click the Groups tab.
2. Choose [New Record].
3. Enter the name of the include group in the Value field. This name is a tag for internal use of the DHCP server only. For more information see Section 7.5.3.4.1.
4. Choose Member of Group (optional). Enter the name of an include group that the include group is joining. Use this feature to create hierarchies of groups and minimize duplication elsewhere.
5. Choose Group Members (optional).

Enter the names of subnets, nodes, or other groups that are to be members of the group, that will pull in this group's parameters. If you have already created a node or subnet group or groups that are members of the include group you are entering the DHCP GUI will display the names of these groups in the Group Members field. If you want to include a new member or delete an existing member you can do so here in the group's Group Members field or under the Subnets or Nodes tab in the Member of Group field of the groups pulling in this group.

6. Set up key information:

   Enter the vendor class (for example, TCPVMS or JOIN) that identifies the DHCP client vendor class to which this entry should apply. Note that you can have multiple include groups with the same name if they have different vendor class key values. If the entry should apply to any vendor class or you are not using vendor classes leave the vendor class field blank.

7. Choose from the lists of DHCP parameters on the drop-down list.

   Different lists of DHCP parameters are available on the drop down list. Choose either BASIC DHCP Parameters or DHCP Parameters.

8. As appropriate, enter information for Network, Lease, Time, BOOTP, NetBIOS, X Window, TCP, IP, and Link parameters. For more information about these parameters, refer to Section 7.5.3.4.

9. Choose Update from the File menu.

The new configuration takes effect immediately.

7.5.3.3.3 Removing a group record To remove a group record:

1. Click the Groups tab.
2. Choose DHCP Parameters from the drop-down list.
3. Choose the Group record you want to delete.
4. Click Delete.

Changes to the database take effect immediately.

7.5.3.4 Nodes, Subnets, Group Parameters

This section describes the subnet, group, and node parameters. The parameters are grouped by the following categories:

- Name and ID Parameters
- Key Parameters
- BOOTP Parameters
- IP Parameters
- Lease Parameters
- Link Parameters
- NetBIOS Parameters
- Network Parameters
- TCP Parameters
- Time Parameters

- X Window Parameters

For any parameter, list the values in order of preference.

7.5.3.4.1 Name/ID parameters Name and identification parameters determine the name of the configuration and information that identifies which client or clients are being configured by this record.

Name
Specifies the name for this subnet, node, or include group configuration. The names used here are tags for the internal use of the DHCP server. You can name them as you choose but do not use the same name more than once except where you use a different vendor class for the duplicate names.

Group Members
Specifies the names of subnet, node, and include groups that are members of the group (that is, those that inherit this group's parameters).

Member of Group
Specifies the name of the group that the subnet, node, or include group is joining.

7.5.3.4.1.1 Limitations on Group Hierarchies

The hierarchies provided for with member groups do not support multiple inheritance. An include group can have multiple members, but an include, subnet, or node group can be a member of only one group. For example, you can make Group_A with members Group_B and Group_C, but you can not make Group_A a member of Group_B and Group_C.

7.5.3.4.2 Key Parameters Key parameters identify the keys for the configuration record. The Key parameters include Hardware Address/Client ID, Hardware Type, Net or Subnet IP Address, and Vendor Class.

Hardware Address/Client ID
This parameter specifies the hardware address (MAC address) of the node. Enter the hardware address in the format *xx:xx:xx:x:xx:xx*, for example, 00:08:C7:08:E3:63. The hardware address is assigned during manufacturing and usually is displayed when you turn on or reboot your workstation.

Hardware Type
This field takes a string of characters and specifies the network type associated with this node, such as Ethernet or token ring.

Enter either the symbol or the actual number as shown in Table 7–6. For example, to specify Ethernet as the hardware type, enter either the symbol ether or the number 1.

Net or Subnet IP Address
Specifies the address of the subnet record (if its a Subnet configuration record). Enter the IP address that identifies this subnet portion of the network, for example, 129.84.3.0.

Vendor Class
A DHCP client can pass a vendor class string to the server to identify the client vendor implementation. For example, TCPVMS for the TCP/IP Services DHCP client. The DHCP server uses the vendor class string as part of the key lookup when determining which groups of configured parameters apply to the client. The information is a string of octets, usually ASCII, that the server interprets.

7.5.3.4.3 BOOTP Parameters The server version of DHCP fully supports the following BOOTP parameters. If a BOOTP client makes a request of the server, it acts as a BOOTP server.

Boot File
Specifies the fully qualified path name of the client's default boot image.

Boot File Server Address
Specifies the server address of the boot file.

Boot File Server Name
Specifies the host name of the server with the boot file.

Boot File Size
Specifies the length in 512-octet blocks of the default boot image for the client. Specify the file length as a number.

Cookie Servers
Specifies a list of RFC 865 cookie servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

DNS Domain Name
Specifies the domain name the client should use when resolving host names through the Domain Name System.

DNS Servers
Specifies a list of DNS (STD 13, RFC 1035) name servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

Extensions Path
Specifies a string which identifies a file, retrievable through TFTP, that contains information that the server can interpret in the same way as the 64-octet vendor-extension field in the BOOTP response. There is no limit on the length of this file.

Home Directory
Specifies the directory where the boot file resides, if it is not specified in the boot file name.

Also specifies the name of the client. The name can or can not be qualified with the local domain name. See RFC 1035 for character-set restrictions.

Host IP Address (BOOTP only)
Specifies the host IP address for BOOTP clients.

Host Name
Specifies the host name parameter if you are setting up a configuration for a single client identified by its MAC address.

Also specifies the name of the client. The local domain name can or can not qualify the client name. See RFC 1035 for character-set restrictions.

IEN-116 Name Servers

Specifies a list of IEN-116 name servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

Impress Servers

Specifies a list of Imagen Impress servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

Log Servers

Specifies a list of MIT-LCS UDP log servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

LPR Servers

Specifies a list of RFC 1179 line-printer servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

Merit Dump File

Specifies the path name of a file to which the client's core image should be dumped in the event the client fails. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

Resource Location Servers

Specifies a list of RFC 887 resource location servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

Root Path

Specifies the path name that contains the client's root directory or partition. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

Routers

Specifies the list of IP addresses for routers (gateways) on the client's subnet. If you specify a default gateway of 0.0.0.0, the server uses the client's IP address as the default gateway address.

Use this format: *ddd.ddd.ddd.ddd*.

Subnet Mask

Specifies the client's subnet mask as described in RFC 950.

A subnet mask allows the addition of subnetwork numbers to an address, and provides for more complex address assignments.

If you specify both the subnet mask and the router option in a DHCP reply, the subnet mask option must be first.

Use this format: *ddd.ddd.ddd.ddd*.

Send Client's Host Name

Specifies whether the server should send the client's host name to the client in the reply.

The following are valid values:

False: Do not send the client's host name. Default.

True: Send the client's host name.

Swap Server

Specifies the IP address of the client's swap server.

Use this format: *ddd.ddd.ddd.ddd*.

TFTP Root Directory

Specifies the root directory for Trivial File Transfer Protocol (TFTP).

Time Offset

Specifies the offset of the client in seconds from Universal Coordinated Time (UTC).

Time Servers

Specifies a list of RFC 868 time servers available to the client. Enter the servers in order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

Vendor Magic Cookie

Specifies a vendor magic cookie for the client.

7.5.3.4.4 IP parameters IP layer parameters affect the operation of the IP layer on a per-host basis.

Broadcast Address

Specifies the broadcast address in use on the client's subnet.

Forward Nonlocal Datagrams

Specifies whether the client should configure its IP layer to allow forwarding of datagrams with nonlocal source routes.

The following are valid values:

False: Disable forwarding of datagrams with nonlocal source routes.

True: Enable forwarding.

IP Forwarding

Specifies whether the client should configure its IP layer for packet forwarding.

The following are valid values:

False: Disable IP forwarding.

True: Enable IP forwarding.

IP Time-to-Live

Specifies the default time-to-live that the client should use on outgoing datagrams. Specify time-to-live as an octet.

Minimum value 1

Maximum value 255

Interface MTU

Specifies the maximum transmit unit (MTU) to use on this interface. Specify the MTU as a 16-bit unsigned integer.

Minimum legal value is 68.

Maximum Reassembly Size

Specifies the maximum size datagram that the client should be prepared to reassemble. Specify the size as a 16-bit unsigned integer.

Minimum legal value is 576.

MTU Plateaus

Specifies a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is formatted as a list of 16-bit unsigned integers, ordered from smallest to largest.

The minimum value cannot be smaller than 68.

PMTU Timeout

Specifies the timeout to use when aging Path MTU values discovered by the mechanism defined in RFC 1191. Specify the timeout in seconds as a 32-bit unsigned integer.

Perform Mask Discovery

Specifies whether the client should perform subnet mask discovery using the Internet Control Message Protocol (ICMP).

The following are valid values:

False: Client should not perform mask discovery.

True: Client should perform mask discovery.

Perform Router Discovery

Specifies whether the client should solicit routers using the Router Discovery mechanism defined in RFC 1256.

The following are valid values:

False: Client should not perform router discovery.

True: Client should perform router discovery.

Policy Filters

Specifies policy filters for nonlocal source routing. The filters consist of a list of IP addresses and masks that specify destination/mask pairs with which to filter incoming source routes.

The client should discard a source-routed datagram whose next-hop address does not match one of the filters.

Use this format: *ddd.ddd.ddd.ddd*.

Solicit Router

Specifies the IP address to which the client should transmit router solicitation requests.

Use this format: *ddd.ddd.ddd.ddd*.

Static Routes

Specifies a list of static routes that should be installed in the client's routing table. If you specify multiple routes to the same destination, list them in descending order of priority.

The routes consist of a list of IP address pairs. The first address is the destination address, and the second address is the router for the destination.

Note

The default route (0.0.0.0) is an illegal destination for a static route.

Use this format: *ddd.ddd.ddd.ddd*.

Subnets Are Local

Specifies whether the client can assume that all subnets of the IP network to which the client is connected use the same MTU (maximum transmit unit) as the subnet of the network to which the client is directly connected.

The following are valid values:

False: The client should assume that some subnets of the directly connected network can have smaller MTUs.

True: All subnets share the same MTU.

Supply Masks

Specifies whether the client should respond to subnet mask requests using the Internet Control Message Protocol (ICMP).

The following are valid values:

False: Client should not respond.

True: Client should respond.

7.5.3.4.5 Lease parameters Lease parameters allow you to change information about the IP lease times. Lease times determine the length of time a client can use an IP address.

DHCP Rebinding Time

Specifies the time interval in seconds from address assignment until the client requests a new lease from any server on the network.

DHCP Renewal Time

Specifies the time interval in seconds from address assignment until the client attempts to extend the duration of its lease with the original server.

DHCP Lease Time

The client uses this option in a client request (DHCPDISCOVER or DHCPREQUEST) message to request a lease time for the IP address.

The server uses this option in a server reply (DHCPOFFER) message to specify the lease time it is willing to offer.

Enter the time in months, days, hours, minutes, and seconds; for example, 2 months 5 days 45 minutes. By default, the server interprets the lease in seconds. For an infinite lease for a BOOTP client, specify a -1.

7.5.3.4.6 Link Parameters Link Layer parameters affect the operation of the Link layer on a per-host basis.

ARP Cache Timeout
Specifies the timeout in seconds for ARP cache entries. The time is specified as a 32-bit unsigned integer.

Ethernet Encapsulation
If it is an Ethernet interface, use this option to specify whether the client should use Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation.

The following are valid values:

False: Use RFC 894 encapsulation.

True: Use RFC 1042 encapsulation.

Trailer Encapsulation
Specifies whether the client should negotiate the use of trailers (RFC 893) when using the ARP protocol.

The following are valid values:

False: Client should not attempt to use trailers.

True: Client should attempt to use trailers.

7.5.3.4.7 NetBIOS Parameters NetBIOS parameters configure NetBIOS related parameters on a per-host basis.

NetBIOS Datagram Distribution Server
Specifies a list of RFC 1001/1002 NBDD servers listed in the order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

NetBIOS Name Server/WINS Server
Specifies a list of RFC 1001/1002 NBNS name servers listed in the order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

NetBIOS Node Type
Allows you to configure NetBIOS-over-TCP/IP clients as described in RFC 1001/1002. Specify the value as a single octet (from 0 to 255) that identifies the client type as shown in Table 7–7.

Table 7–7 NetBIOS Node Type and Value

Node Type	Value (hexadecimal)
B-node	1
P-node	2
M-node	4
H-node	8

Note

The NetBIOS over TCP/IP clients must be configurable.

NetBIOS Scope
The NetBIOS scope option specifies the NetBIOS scope text parameter for the client as specified in RFC 1001/1002. There can be character-set restrictions.

7.5.3.4.8 Network Parameters Network parameters allow you to change basic network configuration information.

Finger Servers
Specifies a list of finger servers available to the client. List the servers in the order of preference.

IRC Servers
Specifies a list of IRC (Internet Relay Chat) servers available to the client. List the servers in the order of preference.

Mobile IP Home Agents
Specifies a list of IP addresses indicating mobile IP home agents available to the client. List the agents in the order of preference.

NNTP Servers
Specifies a list of Network News Transfer Protocol (NNTP) servers available to the client. List the servers in the order of preference.

NetWare Domain
Specifies the NetWare domain name.

NetWare Options
Specifies a list of NetWare servers.

POP3 Servers
Specifies a list of Post Office Protocol 3 (POP3) servers available to the client. List the servers in the order of preference.

SMTP Servers
Specifies a list of Simple Mail Transfer Protocol (SMTP) servers available to the client. List the servers in the order of preference.

STDA Servers
Specifies a list of StreetTalk Directory Assistance (STDA) servers available to the client. List the servers in the order of preference.

StreetTalk Servers
Specifies a list of StreetTalk servers available to the client. List the servers in the order of preference.

WWW Servers
Specifies a list of World Wide Web servers available to the client. List the servers in the order of preference.

7.5.3.4.9 TCP Parameters TCP parameters affect the operation of the TCP layer on a per-host basis.

Keep Alive Interval
Specifies the interval that the client should wait before sending a keepalive message on a TCP connection.

A value of 0 (zero) indicates that the client should not generate keepalive messages on connections unless an application requests them.

Specify the time in seconds as a 32-bit unsigned integer.

Keep Alive Octet
This parameter specifies whether the client is to send TCP keepalive messages with a garbage octet for compatibility with older implementations.

The following are valid values:

False: Do not send a garbage octet.

True: Send a garbage octet. (Sets the compatibility mode.)

TCP Default Time-to-Live
This option specifies the default time-to-live that the client uses when sending TCP segments.

Minimum value is 1.

7.5.3.4.10 Time Parameters Time parameters allow you to change information about network time services available to clients on the network.

Network Time Protocol (NTP) Servers
Specifies a list of RFC 1305 time servers available to the client. List the server addresses in the order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

7.5.3.4.11 X Window Parameters X parameters configure X11-related parameters on a per-host basis.

X Window Display Manager
Specifies a list of IP addresses of systems that are running the X Window System display manager and that are available to the client.

Enter IP addresses in the order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

X Window Font Server
Specifies a list of X Window System font servers available to the client. Enter the server addresses in the order of preference.

Use this format: *ddd.ddd.ddd.ddd*.

7.6 Configuring DHCP/BOOTP IP Addressing

After you convert your existing BOOTP file to the new DHCPCAP. file as described in Section 7.4.1, you are ready to begin serving your existing BOOTP clients without any further changes.

This section explains how to use the GUI to configure static IP addressing for any DHCP/BOOTP clients you add in the future, as appropriate.

Configuring static IP addressing for DHCP and BOOTP client requires different steps described in the following sections.

7.6.1 Static IP Addressing for BOOTP Clients

To define static IP addressing, specify a specific IP address for a specific MAC address as follows:

1. Start the GUI by entering the following command:

   ```
   $ DHCPGUI
   ```

2. Click the Nodes tab.
3. Choose [New Record].
4. Enter the host name (Name).
5. Enter the MAC/hardware address. For example, 08:00:20:3f:12:4b.
6. Choose Hardware Type from Key Parameters. Enter the type of network on which the node resides. Enter the hardware type using the symbol or the type number as shown in Table 7–6.
7. Choose [Host IP Address].
8. Enter the Host IP address of the host computer for this node.
9. As appropriate, enter information for Network, Lease, Time, BOOTP, NetBIOS, X Window, TCP, IP, and Link parameters. For more information about these parameters, refer to Section 7.5.3.4.
10. Choose Update from the File menu to update the server with the new configuration.

7.6.2 Static IP Addressing for DHCP Clients

Select static addressing if you want to assign a specific IP address with a permanent lease time to a DHCP client, and you do not want the client to be able to release this IP address. Also, select static addressing if you need to select an IP address that is not part of any IP address pool.

Selecting an IP address from outside an IP address pool allows the server to specify a permanent mapping between a DHCP client's MAC address and the desired IP address. A client can reuse and release any address within an IP pool.

To configure a specific, permanent address for a DHCP client, do the following:

1. Start the GUI by entering the following command:

   ```
   $ DHCPGUI
   ```

2. Click the Server/Security tab.
3. Choose Active IP Snapshot, then choose [New Record].
4. Enter the MAC address.
5. Enter the MAC type.
6. Enter the MAC address length.
7. Enter an IP address that does not belong to any IP address pools.
8. Enter -1 (infinite lease) for the lease expiration.
9. Enter the server IP address.
10. If you want a name associated with the client, specify the client's host name and domain name.

If you set the Use MAC addr as Client ID parameter to True, the server uses the MAC address to uniquely identify the clients. The MAC address field might not be the actual MAC address of the client's network adapter. Clients that modify the structure of the MAC address before sending it to the server include:

- Windows 95, Windows NT, and Windows for Workgroups 3.11 with Microsoft TCP/IP

  On these platforms, the MAC address is prefixed with the hardware type. The MAC type is 0 and the length is 7 (instead of 6). For example, if your Ethernet address is 11:22:33:44:55:66, you need to specify the following for the static IP mapping:

  – MAC Address: 01:11:22:33:44:55:66

  – MAC Type: 0

  – MAC length: 7

- FTP Software's OnNet client

  On this platform, the string "cid-" prefixes the MAC address. The MAC type is 0 and the length is 16. For example, if your Ethernet address is 11:22:33:44:55:66, you need to specify the following for the static IP mapping:

  – MAC Address: cid-112233445566

  – MAC Type: 0

  – MAC length: 16

7.7 Configuring DHCP Manually

After you run the TCPIP$CONFIG.COM procedure and enable the DHCP server on your system, you can manually define the following client information on a case-by-case basis:

- Static, dynamic, or finite addressing
- Other identifying parameters, such as default gateways and DNS domain names

7.7.1 Tasks Involved

Defining client addressing and additional parameters manually involves the following steps:

1. Modify the appropriate text-based configuration files.

   These files are listed in Section 7.2.2.

   You manually edit the DHCP configuration files using a text editor such as EDT, TPU, or LSE. Depending on your environment, you may or may not need to modify all the files.

2. If appropriate, run DHCP utilities to update the binary databases.

   When you are modifying information already stored in the databases, you use command line utilities to access and update the database contents. These utilities are defined as both OpenVMS and UNIX commands. Table 7–10 lists the utilities.

3. Reinitialize the DHCP server for the changes to take effect (see Section 7.7.3).

7.7.2 Modifying the Client Configuration Parameters File

The DHCPCAP. file describes the various configuration parameters for the clients. This file is similar to the standard `bootptab` file used by most BOOTP servers. Each entry in the file can describe a single machine (per-node basis) or all the machines within a subnet (per-subnet basis) or a group of machines (per-group basis).

7.7.2.1 DHCPCAP Configuration Syntax

The DHCPCAP. configuration file uses two-character, case-sensitive symbols that represent host parameters. Colons (:) follow and separate parameters from one another. For example, `gw` specifies gateway. For a list of the available symbols, see Section 7.7.2.5.

The following is the format of a configuration file entry:

entryname:symbol=value:symbol=value:symbol=value:

In this format:

- *entryname* is usually the name of the BOOTP or DHCP client.
- *symbol* is the two-character symbol that describes the parameters to be associated with the client.
- *value* is a valid entry that represents the symbol. For more information, see Section 7.7.2.4.

Example 7–8 shows a sample DHCPCAP file entry.

Example 7–8 Sample Single-Host DHCPCAP File Entry

```
mypc:\
    :ht=ether:\
    :ha=112233445566:\
    :ip=143.32.3.10:\
    :gw=143.32.3.1:\
    :dn=acme.com:
```

7.7.2.2 DHCPCAP Configuration Rules

When you create the DHCPCAP file, entries must conform to the following rules:

- Start each new host entry on a new line. You can make a single entry span multiple lines by ending each line with a backslash (\).
- Terminate each entry name and each symbol/value pair with a colon (:). For readability, you can leave blank spaces between symbol/value pairs.
- Enter the entry name in the first field in the configuration file entry.
- Make sure that the host hardware type (`ht`) precedes the host hardware address (`ha`).

You can delete symbol values associated with a particular client by entering an at sign (@) immediately following the symbol. For example, gw@.

Both BOOTP and DHCP interpret lines that begin with any of the following as comments:

- The pound sign (#)
- One or more blank spaces followed by #

- A blank line

7.7.2.3 DHCPCAP Configuration Examples

Example 7–9 shows a sample single-host DHCPCAP. file entry. This entry, mypc, describes the configuration for a BOOTP client. It describes the client itself, its IP address, the default gateway, and the domain name.

Example 7–9 Sample Single Host DHCPCAP Entry

```
mypc:\
    :ht=ether:\
    :ha=112233445566:\
    :ip=143.32.3.10:\
    :gw=143.32.3.1:\
    :dn=acme.com:
```

Example 7–10 shows a subnet DHCPCAP. file entry. This entry, subnet5, describes the parameters for all the clients on a particular subnet, 143.32.5.0. It describes the default gateway, subnet mask, domain name, DNS server address, and lease time of the address.

Example 7–10 Sample Subnet DHCPCAP Entry

```
subnet5:\
       :nw=143.32.5.0:\
       :gw=143.32.5.1:\
       :sm=255.255.255.0:\
       :dn=engr.acme.com:\
       :ds=143.32.5.10:\
       :lt=3600:
```

7.7.2.4 Symbol Value Formats

The symbol values require specific formats. Use only the following formats:

- ASCII string

  Enclose this string in quotation marks (“string”) if it contains any of the special characters: colon (:), pound sign (#), tab, or space.

- ASCII integer list

  A list of integers separated by white space consisting of ASCII-format characters that represent an unsigned hexadecimal, octal, or decimal integer.

  - Begin the string with 0X or 0x if this is a hexadecimal integer.
  - Begin the string with zero (0) if this is an octal integer.

- IP address list

  ASCII string representing an IP address in dotted-decimal notation (for example, 128.119.95.2).

  An IP address list is a string of one or more IP addresses, with the addresses separated by spaces. For example:

  ```
  tg=128.119.91.2   128.119.95.42    128.119.95.8
  ```

  You can also use IP address lists to define DHCP address ranges, routing policy filters, and static routes.

- ASCII-format representation of a hexadecimal integer that DHCP and BOOTP interpret as a hardware address.

The ASCII string must have the correct number of digits for the specified hardware type; for example, twelve digits for a 48-bit Ethernet address. To improve readability, you can:

- Separate the two-digit sequences (bytes) with hyphens (-).
- Separate the two-digit sequences (bytes) with periods (.).
- Add a 0x prefix to each byte (or only some bytes) of the address.
- Add a hyphen between some bytes and 0x prefixes before others.
- Add a period between some bytes and 0x prefixes before others.

Examples of valid hexadecimal ASCII strings are:

```
ha=7F-FF-81-00-0A-47
ha=0X7F0XFF0X81000A47
ha=0X7F-FF0XF8-1000A47
```

- Booleans and switches

  A boolean symbol performs a function just by its presence. A switch is the value 0 or 1, and it associates one of two functions to those values (usually, disable and enable, respectively).

```
:ms=1:\              #This is an example of switch type field
:hn:\                #This is an example of a boolean type field
```

7.7.2.5 DHCP Configuration Symbols

Table 7–8 describes each DHCP configuration file symbol and indicates whether you use the symbol in DHCP configuration only or in BOOTP and DHCP configuration.

Table 7–8 BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
as	Maximum datagram reassembly size	ASCII integer	Specifies the maximum size datagram that the client should be prepared to reassemble. The minimum value is 576.
at	ARP cache timeout	ASCII integer	Specifies the timeout (in seconds) for ARP cache entries.
ba	Broadcast address	IP address	Specifies the broadcast address in use on the client's subnet.
bf	Boot file	ASCII string	Specifies the fully qualified path name of the client's default boot image.
br	IP forwarding	Boolean	Specifies whether the client should configure its IP layer for packet forwarding. A value of 0 means disable IP forwarding, and a value of 1 means enable IP forwarding.
bs	Boot file size	ASCII integer or auto	Specifies the length in 512-octet blocks of the default boot image for the client.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
bw	NetBIOS name servers	IP address list	Specifies a list of RFC 1001/1002 NBNS name servers listed in order of preference.
bx	NetBIOS over TCP/IP datagram distribution server	IP address list	Specifies a list of RFC 1001/1002 NBDD servers listed in order of preference.
by	NetBIOS over TCP/IP node type	ASCII integer	Specifies whether clients can be configured as described in RFC 1001 and 1002. The NetBIOS node type option allows NetBIOS over TCP/IP configurable clients to be configured as described in RFC 1001 and 1002. Specify the value as a single octet (from 0 to 255) that identifies the client type.
bz	NetBIOS over TCP/IP scope	ASCII string	Specifies the NetBIOS over TCP/IP scope text parameter for the client as specified in RFC 1001/1002. There can be character-set restrictions.
ck	Client identifier	Opaque	
cs	Cookie server address list	IP address list	Specifies a list of RFC 865 cookie servers available to the client. Enter servers in order of preference.
ct	Vendor class	String	Specifies the vendor type and configuration of a DHCP client. The information is a string of *n* octets, interpreted by servers. Vendors may choose to define specific vendor class identifiers to convey particular configuration or other identification information about a client. For example, the identifier may encode the client's hardware configuration. Servers not equipped to interpret the class-specific information sent by a client must ignore it (although it may be reported).
da	STDA servers	IP address list	Specifies a list of StreetTalk Directory Assistance (STDA) servers available to the client. Servers should be listed in order of preference.
df	Merit dump file	ASCII string	Specifies the path name of a file to which the client's core image should be dumped in the event the client fails. The path is formatted as a character string consisting of characters from the NVT ASCII character set.
dn	DNS domain name	ASCII String	Specifies the domain name that the client should use when resolving host names via the Domain Name System.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
ds	DNS servers	IP address list	Specifies a list of Domain Name System (RFC 1035) name servers available to the client. Enter servers in order of preference.
ec	Ethernet encapsulation	0 or 1	Specifies whether the client should use Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation if the interface is an Ethernet. The switch values are: • 0 - Use RFC 894 encapsulation • 1 - Use RFC 1042 encapsulation
ef	Extensions path	ASCII string	Specifies a file, retrievable through TFTP, that contains information that can be interpreted in the same way as the 64-octet vendor-extension field in the BOOTP response. The length of the file is unconstrained.
fi	Finger servers	IP address list	Specifies a list of finger servers available to the client. Servers should be listed in order of preference.
fn	Forward nonlocal datagarams	0 or 1	Specifies whether the client should configure its IP layer to allow forwarding of datagrams with nonlocal source routes.
gw	Gateway address list	IP address list	Specifies a list of the IP addresses of gateways for the specified subnet. This list consists of the default routes.
ha	Client's hardware address	ASCII string	Specifies whether host names can be assigned by the hardware address. If so specified, the client host, provided it remains in the same domain, retains the same name, even if its IP address changes.
hn	Host name	Boolean	Specifies that the DHCP server should write the client's host name to the vend field of the DHCP reply packet and send the packet to the client. Can appear only in the format hn: or hn@:.
ho	Host name	ASCII string	Specifies the name of the client. The name may or may not be qualified with the local domain name.
ht	Client's hardware type	ASCII string or ASCII integer	Specifies the hardware type code as assigned in the ARP section of RFC 1340, Assigned Numbers.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
hr	Forwarding enable/disable	0 or 1	Specifies whether the client should configure its IP layer for packet forwarding. The values are: • 0 - Disable • 1 - Enable
im	Impress server address list	IP address list	Specifies a list of Imagen Impress servers available to the client. Enter servers in order of preference.
ip	Client IP address	IP address	Specifies the IP address of the BOOTP client or a single IP address to assign the DHCP client.
it	IP time to live	ASCII string	Specifies the default time to live that the client should use on outgoing datagrams.
ki	TCP keepalive interval	ASCII integer	Specifies the interval (in seconds) that the client TCP should wait before sending a keepalive message on a TCP.
ko	TCP keepalive garbage	0 or 1	Specifies whether the client should send TCP keepalive messages with an octet of garbage for compatibility with older implementations.
lg	Log server	IP address list	Specifies a list of MIT-LCS UDP log servers available to the client. Enter servers in order of preference.
lp	LPR server address list	IP address list	Specifies a list of RFC 1179 line printer servers available to the client. Enter servers in order of preference.
lt	Lease time	ASCII integer	Specifies in a client request, that a client is allowed to request a lease time for the IP address. In a server reply, specifies the lease time the server is willing to offer. Enter the time in seconds.
md	Perform mask discovery	0 or 1	Specifies whether the client should perform subnet mask discovery using ICMP.
mm	Maximum DHCP message size	Integer	Specifies the maximum length DHCP message that it is willing to accept. The length is specified as an unsigned 16-bit integer. A client may use the maximum DHCP message size option in DHCPDISCOVER or DHCPREQUEST messages, but should not use the option in DHCPDECLINE messages.
ms	Mask supplier	0 or 1	Specifies whether the client should respond to subnet mask requests using ICMP.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
nn	NNTP	IP address list	Specifies the NNTP server.
no	NetWare options	Opaque	
ns	IEN-116 name server address list	IP address list	Specifies a list of IEN 116 name servers available to the client. Enter servers in order of preference.
nt	NTP servers	IP address list	Specifies a list of NNTP (Network Time Protocol) servers.
ov	Overload file/sname	Integer	Specifies that the DHCP `sname` or `file` fields are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allotted for options.
pf	Policy filter	IP address list	Specifies policy filters for nonlocal source routing. The filters consist of a list of IP addresses and masks that specify destination/mask pairs with which to filter incoming source routes.
pl	Path MTU plateau table	ASCII integer list	Specifies a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The minimum value is 68.
pt	Path MTU aging timeout	Integer	Specifies the timeout (in seconds) to use when aging Path MTU values are discovered by the mechanism defined in RFC 1191 [12]. The timeout is specified as a 32-bit unsigned integer.
rd	Perform router discovery	0 or 1	Specifies whether the client should solicit routers using the Router Discovery mechanism defined in RFC 1256.
rl	Resource location protocol server address list	IP address list	Specifies a list of RFC 887 Resource Location servers available to the client. Servers should be listed in order of preference.
rp	Root path	ASCII string	Specifies the path name that contains the client's root directory or partition. The path is formatted as a character string consisting of characters from the NVT ASCII character set.
rs	Router solicitation address	IP address	Specifies the address to which the client should transmit router solicitation requests.
sa	Boot server address	IP address	Specifies the IP address of the TFTP server the client uses.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
sl	All subnets are local	0 or 1	Specifies whether the client can assume that all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly connected.
sn	Boot file server name	ASCII string	Specifies the host name of the bootfile server.
sm	Subnet mask	IP address	Specifies the client's subnet mask as per RFC 950. A subnet mask allows the addition of subnetwork numbers to an address and provides more complex address assignments. If both the subnet mask and the router option are specified in a DHCP reply, the subnet mask option must be first.
sp	SMTP servers	IP address list	Specifies a list of SMTP (Simple Mail Transport Protocol) servers available to the client. Servers should be listed in order of preference.
sr	Static route	IP address list	Specifies a list of static routes that the client should install in its routing cache. If multiple routes to the same destination are specified, they are listed in descending order of priority. The routes consist of a list of IP address pairs. The first address is the destination address, and the second address is the router for the destination.
st	StreetTalk servers	IP address list	Specifies a list of StreetTalk servers available to the client. Servers should be listed in order of preference.
sw	Swap server	IP address	Specifies the IP address of the client's swap server.
sv	Server IP	IP address	Specifies the server ID in a DHCOFFER and DHCPREQUEST message and optionally in a DHCPACK and DHCPNAK messages. DHCP servers include this option in the DHCPOFFER in order to allow the client to distinguish between lease offers. DHCP clients use the contents of the "server identifier" field as the destination address for any DHCP messages unicast to the DHCP server. DHCP clients also indicate which of several lease offers is being accepted by including this option in a DHCPREQUEST message.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
t1	DHCP renewal time	Integer	Specifies the time interval (in seconds) from address assignment until the client transitions to the RENEWING state. The value is specified as a 32-bit unsigned integer.
t2	DHCP rebinding time	Integer	Specifies the time interval (in seconds) from address assignment until the client transitions to the REBINDING state. The value is specified as a 32-bit unsigned integer.
to	Time offset	ASCII integer or auto	Specifies (in seconds) the offset of the client's subnet in seconds from Coordinated Universal Time (UTC). The offset is expressed as a twos complement 32-bit integer. A positive offset indicates a location east of the zero meridian and a negative offset indicates a location west of the zero meridian.
tr	Trailer encapsulation	0 or 1	Specifies whether the client should negotiate the use of trailers (RFC 893) when using the ARP protocol.
tu	Interface MTU	ASCII integer	Specifies the MTU to use on this interface.
ts	Time server address list	IP address list	Specifies a list of RFC 868 time servers available to the client. Servers should be listed in order of preference.
tt	TCP default TTL	ASCII integer	Specifies the default time to live that the client should use when sending TCP segments.
uc	User class	ASCII string	Specifies the type or category of user or application the client represents. This option is used by a DHCP client to optionally identify the type or category of user or application it represents. The format of this option is an NVT ASCII text object of varying length which represents a user class of which the client host is a member. DHCP administrators may define specific user class identifiers to convey information about a host's software configuration or about its user's preferences. For example, an identifier may specify that a particular DHCP client is a member of the class "accounting auditors", which have special service needs such as a particular database server.

(continued on next page)

Table 7–8 (Cont.) BOOTP/DHCP Configuration File Symbols

Symbol	Function	Value Format	Description
vm	Vendor's magic cookie selector	ASCII string	Specifies a vendor magic cookie for the client.
xd	X Window System display manager	IP address list	Specifies a list of IP addresses of systems that are running the X Window System display manager that are available to the client. Enter addresses in order of preference.
xf	X Window System font server	IP address list	Specifies a list of X Window System font servers available to the client. Enter addresses in order of preference.
yd	NIS domain name	ASCII string	Specifies the name of the client's NIS domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.
ys	NIS servers	IP address list	Specifies a list of IP addresses indicating NIS servers available to the client. Servers should be listed in order of preference.
zd	NIS+ domain name	ASCII string	Specifies the name of the client's NIS+ domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.
zs	NIS+ server	IP address list	Specifies a list of IP addresses indicating NIS+ servers available to the client. Servers should be listed in order of preference.

Table 7–9 Vendor Specific Options

Symbol	Function	Value Format	Description
For Join DHCP clients:			
cb	Client binary	ASCII string	Refer to the JOIN documentation at the following URL: http://www.join.com/doc/
mf	NFS mounted file systems	ASCII string list	Refer to the JOIN documentation at the following URL: http://www.join.com/doc/
pr	Printers	ASCII string list	Refer to the JOIN documentation at the following URL: http://www.join.com/doc/
ps	SVR4 printer setup	ASCII string list	Refer to the JOIN documentation at the following URL: http://www.join.com/doc/
ss	Name service switch	ASCII string	Refer to the JOIN documentation at the following URL: http://www.join.com/doc/

(continued on next page)

Table 7–9 (Cont.) Vendor Specific Options

Symbol	Function	Value Format	Description
For OpenVMS DHCP clients:			
sd	SMTP substitute domain	ASCII string	
sg	SMTP gateway	ASCII string	
sn	SMTP substitute domain not local	Boolean	
sz	SMTP zone	ASCII string	
For SUN DHCP clients:			
aa	Sun Vendor Option #2	IP address list	Refer to the JOIN documentation at the following URL: `http://www.join.com/doc/`

7.7.3 Reinitializing the DHCP Server

Once you have made changes to the configuration files, you must force the server to read them again by sending it an HUP signal (see Section 7.3.1). Enter the following command:

```
$ DHCPSIGHUP
```

7.8 Supporting Utilities

The commands you use to modify and look at the contents of the DHCP databases are described in Table 7–10. TCP/IP Services supplies UNIX type commands for users familiar with the JOIN implementation of a DHCP server.

Table 7–10 DHCP Utility Commands Associated with Databases

DHCP GUI	OpenVMS Command	UNIX Command	Description
Active IP Snapshot Add/Delete	DHCPDBMOD	jdbmod	Modifies lease and naming information in the database. Allows you to preassign static IP addresses to clients. Also allows you to create, delete, or modify existing entries.
Preload MAC Addresses	DHCPDBREG	jdbreg	Populates the database with MAC addresses of known clients. Each record to be loaded is terminated by a new line, and the fields within each record are separated by the vertical bar (|) character.
	DHCPDBDUMP	jdbdump	Reads and outputs information stored in the lease database files including MAC address information, IP addresses, and lease information. Each line of output describes the lease information for one client.
Active IP Snapshot	DHCPSHOWDBS	SHOWDBS	Reads the same information described for DHCPDBDUMP, except that the output is in a format that is easier to read.
	DHCPDBSHOW	DBSHOW	Displays the contents of a single DHCP binary database.

For information about how to enter the DHCP utility commands, see Sections 7.8.1 through 7.8.3.

7.8.1 Using the DHCPDBDUMP, DHCPSHOWDBS, and DHCPDBSHOW Utilities

The DHCPDBDUMP, DHCPSHOWDBS, and DHCPDBSHOW commands dump the information stored in the lease database files. The dumped lease information includes:

- MAC address
- MAC address type
- MAC address length (octets)
- IP address
- Start of lease (UTC)
- Lease expiration (UTC)
- Time when lease can be extended (UTC)
- Time when host last renewed or acquired this lease (UTC)
- IP address of server owning the lease
- Host name (without domain)
- Domain name

Each line of output describes the lease information for one client. The output is in a format that is used by the DHCPDBMOD utility to modify the lease database.

Note

The DHCPBDUMP, DHCPSHOWDBS, and DHCPDBSHOW commands perform read operations on the database, while DHCPDBMOD performs write operations.

The DHCPDBDUMP, DHCPSHOWDBS, and DHCPDBSHOW commands accept a number of different flags and arguments. Table 7–11 lists some of the more important flags.

Table 7–11 DHCPDBDUMP, DHCPSHOWDBS, AND DHCPDBSHOW Command Flags

Flag	Description
-a	Dumps dates in readable format.
-c	Dumps currently active leases only.
-e	Dumps expired leases only.

The following examples show typical output from the DHCPSHOWDBS, DHCPDBSHOW, and DHCPDBDUMP commands:

```
$ DHCPSHOWDBS
IP address       Owner           Expires            Granted on         Last mod
ified      client id
10.10.2.100      10.10.2.6       01/28/2000 13:50   01/28/2000 13:30   01/28/2000 13:30
   0 01:08:00:2b:e5:2c:44

10.10.2.101      10.10.2.6       01/28/2000 13:52   01/28/2000 13:32   01/28/2000 13:32
   0 01:08:00:2b:bf:7d:bb

10.10.4.100      10.10.2.5       01/21/2000 09:27   01/21/2000 09:07   01/21/2000 09:07
   0 01:08:00:2b:e5:2c:44

IP address       Name
10.10.2.101      gody.compaq.com.
10.10.2.100      sarek12.compaq.com.

$ DHCPDBSHOW a
IP address       owner           expires               granted on            la
st modified         network           client id
10.10.2.101      10.10.2.6       02/14/2000 11:18:10   02/14/2000 10:58:10   02
/14/2000 10:58:10   10.10.2.0         0,7,01:08:00:2b:e5:2c:44
10.10.2.103      10.10.2.6       02/14/2000 11:08:21   02/14/2000 10:48:21   02
/14/2000 10:48:21   10.10.2.0         1,6,08:00:2b:2a:de:1e
10.10.2.100      10.10.2.6       02/14/2000 11:14:23   02/14/2000 10:54:23   02
/14/2000 10:54:23   10.10.2.0         0,7,01:08:00:2b:bf:7d:bb
10.10.4.100      10.10.2.5       01/21/2000 09:27:26   01/21/2000 09:07:26   01
/21/2000 09:07:26   10.10.4.0         0,7,01:08:00:2b:e5:2c:44
10.10.2.104      10.10.2.6       02/14/2000 11:09:33   02/14/2000 10:49:33   02
/14/2000 10:49:33   10.10.2.0         1,6,08:00:2b:2a:de:a8

Record count = 5
```

```
$ DHCPDBDUMP
01:08:00:2b:e5:2c:44|0|7|10.10.2.100|949084208|949085408|949084808|949084208|
   10.10.2.6|sarek12|compaq.com|
01:08:00:2b:bf:7d:bb|0|7|10.10.2.101|949084349|949085549|949084949|949084349|
   10.10.2.6|gody|compaq.com|
01:08:00:2b:e5:2c:44|0|7|10.10.4.100|948463
```

7.8.2 Using the DHCPDBMOD Utility

The DHCPDBMOD command modifies the lease and naming information in the database files. It allows the user to create, delete, or modify existing database entries and to preassign static IP address to clients.

The utility takes input from a file that describes various entries in the database. The syntax of each entry is similar to the output of DHCPDBDUMP.

Use the following format:

- Terminate each record to be loaded by a new line.
- Delimit the fields within each record with the vertical bar (|) character.
- Express date fields in one of the following ways:
  - Coordinated Universal Time (UTC), the number of seconds since 00:00 01/01/1970 GMT
  - A format more easily understood, such as Mon Jan 09 1995 10:00 or 01/09/1995 10:00:00

Example 7–11 shows a sample entry. The first entry describes the client called alpha.acme.com with the IP address 143.32.3.20.

The second entry describes a Microsoft DHCP client with the IP address 143.32.3.21. No name is given for this client.

Example 7–11 Sample DHCPDBMOD Entry

```
$ DHCPDBMOD
00:a0:24:8c:6b:091 |12 |63 |143.32.3.204 |8449894575 |8449894666 |8449894667 |8449894668
    |143.32.3.199 |alpha10|acme.com11|
01:00:40:05:14:df:11|0|7|143.32.3.21|844989457|844989466|844989466|844989466
    |143.32.3.1|||
```

Although some fields can be empty, each entry consists of the following fields:

1 MAC address

2 MAC address type

3 MAC address length (octets)

4 IP address

5 Start of lease (UTC)

6 Lease expiration (UTC)—use -1 to indicate an infinite lease

7 Time when lease can be extended (UTC)

8 Time when host last renewed or acquired this lease (UTC)

9 IP address of server owning the lease

10 Host name (without domain)

11 Domain name

The DHCPDBMOD command accepts a number of different flags and arguments. Table 7–12 shows some of the more important flags.

Table 7–12 DHCPDBMOD Command Flags

Flag	Description
-d	Deletes the record.
-e	Stores the record even if the lease has expired.
-l	Stores the lease information only. Does not store name information.
-n	Stores the name information only. Does not store lease information.
-w	Overwrites the record if a record already exists.

By default, DHCPDBMOD stores both lease and name information for nonexpired and new clients.

7.8.3 Using the DHCPDBREG Utility

Use the DHCPDBREG command to populate the database with the MAC address of known MAC clients. Set the SERVER.PCY parameter Restrict to Known MAC Address to True to use this utility. The DHCPDBREG command can add or remove hosts from the list of known MAC addresses. Use the following syntax when you enter a record:

- Terminate each record to be loaded by a new line.
- Delimit the fields within each record with the vertical bar (|) character.

Each entry contains the following three fields:

- MAC address
- MAC address type
- MAC address length (octets)

The DHCPDBREG command accepts a number of different flags. Two of the most important flags are as follows:

Flags	Description
-d	Deletes the record.
-s	Displays all registered MAC addresses.

7.9 Solving DHCP Server Problems

If the DHCP log file contains the message: “network not administered by server” and you use class A, B, or C IP addressing, check the NETMASKS. file to see that you have entered the netmask correctly for the subnet.

8

Configuring the DHCP Client

DHCP client is the TCP/IP Services component which allows a system to request network configuration information from a DHCP server and then use that information to configure one or more of its network interfaces.

TCP/IP Services DHCP client is an OpenVMS implementation of the Compaq *Tru64* UNIX client.

This chapter reviews key concepts and describes the following topics:

- DHCP client components (Section 8.2)
- DHCP client startup and shutdown (Section 8.3)
- Configuring the DHCP client (Section 8.4)
- TCP/IP management commands (Section 8.5)
- Using the SHOWDHC utility (Section 8.6)

8.1 Key Concepts

When a system connects to a network, in addition to the appropriate network software, it must have configuration information that identifies the system in network communications. As a minimum, it must have an IP address, a broadcast address, and a subnet mask configured before any communication with other systems can take place. This information can be statically configured, that is, permanently stored in a local database and used every time the network is initialized. Or it can be dynamically configured by obtaining the information from a DHCP server during network initialization. The DHCP server maintains the configuration information, and upon a client request for such information, returns the configuration for that particular host through a client and server dialog using the DHCP protocol.

A system can have more than one network interface installed and you can use DHCP client to dynamically configure all or a subset of the installed interfaces. There is one DHCP client process running on a system and it configures all interfaces that are designated as under DHCP control.

In an OpenVMS Cluster, you can use DHCP client to configure one of the systems, a mix of systems or all systems in the cluster. Each system in the cluster using DHCP to configure its interfaces, must run DHCP client.

Note

If a system is running DHCP client, it can not also run a DHCP server.

8.1.1 Designating the Primary Interface

Some of the parameters that are configurable by DHCP are interface specific. Examples of interface-specific parameters are the IP address and subnet mask. Most DHCP configurable parameters, however, are systemwide configurable parameters. Examples of systemwide parameters are the host name and DNS domain name.

The TCP/IP Services DHCP client supports controlled configuration of systemwide configurable items by designation of what is called the primary interface. The primary interface is the interface on which the DHCP client will use systemwide parameters received from the DHCP server to configure the system. Systemwide parameters received on an interface that is not designated primary will not be configured on your system by the DHCP client. There may be only one interface on a system that is designated as the primary DHCP interface, but you are not required to have any interface designated as the primary interface.

For example, if a system has multiple interfaces under DHCP control and the system receives a different host name from a DHCP server on each of the DHCP controlled interfaces, DHCP client uses the host name it receives on the primary interface to configure the host name for the client.

If a system has multiple interfaces and only one is under DHCP control, you can configure the systemwide parameters manually.

DHCP client uses the following rules to resolve conflicts:

- The only-one-primary-interface rule

  This rule solves the potential conflict between two DHCP controlled interfaces on a host getting different systemwide parameter values. To resolve the conflict, you designate one interface to be the primary interface and the parameters values that you receive on that interface are the values DHCP client uses to configure the system. TCP/IP Services does not let you designate two primary interfaces.

- The primary-interface-not-required rule

  This rule solves the problem of DHCP configuring an interface (or interfaces) with an IP address but also keeping manual control of the systemwide parameters. In this case, DHCP client does not designate the interface as the primary interface and ignores any systemwide parameters it receives from a DHCP server.

Systemwide parameters are configured for a system as the last part of processing the final message (a DHCPACK protocol message) from the DHCP server. DHCP client, first configures the interface's IP address, subnet mask, and broadcast address; then, if the interface is designated as the primary interface, DHCP client configures the systemwide parameters.

See Table 8–2 for a list of the DHCP configurable parameters supported by the TCP/IP Services DHCP client.

8.1.2 Requesting a Lease

A DHCP server allocates IP addresses to clients on a temporary or permanent basis. This time period is called a lease. A client can request a lease for some period of time, which the DHCP server can either honor or assign a different time period depending on the policy in force. A client may request a lease for an infinite period of time, but the server may choose to give out a lengthy but not infinite lease. For whatever time period the DHCP server assigns, the DHCP server guarantees not to reassign the IP address to any other system until the lease expires.

Lease times are represented in DHCP dialogs as relative time to be interpreted with respect to the client's clock. If there is drift between the client's clock and the server's clock, the server may consider the lease expired before the client does. To compensate, the server may return a shorter lease duration to the client than the server commits to its local database of client information.

8.1.3 Requesting Parameters

The first service provided by a DHCP server is to provide storage of network parameters for network clients. DHCP clients can query the DHCP server to retrieve the configuration parameters. In its initial discover or request message, a client can supply a list of parameters for which it needs information. If the server does not return any or all of the values for the requested parameters, the client uses default values for any missing values.

8.1.4 Understanding How the DHCP Client Operates

When your system has an interface configured under DHCP control, the following sequence of steps occur at TCP/IP Services startup time:

1. The TCPIP$STARTUP procedure installs the DHCP client image, TCPIP$DHCP_CLIENT.EXE, with the appropriate OpenVMS system privileges.
2. Then it issues the following command to start the interface:

   ```
   $ TCPIP START COMMUNICATION/INITIALIZE
   ```

   This command creates a subprocess and runs the DHCPCONF utility as follows to set up the interface:

   ```
   DHCPCONF -W 30 ifname START
   ```

   Alternatively, the command procedure issues the following command if the interface is the primary interface:

   ```
   DHCPCONF -P -W 30 ifname START
   ```

   The `-W 30` option on the DHCPCONF command line tells DHCPCONF to wait for a maximum of 30 seconds before returning. This wait prevents the TCPIP$STARTUP procedure from hanging indefinitely when there are problems reaching a DHCP server. If the 30-second timer expires, the DHCP client process will, by default, continue to complete the DHCP dialog until it is successful or it is shut down.

3. DHCPCONF creates the DHCP client process.

   If this is the first interface to be configured during the startup procedure, DHCPCONF creates a detached process and runs the TCPIP$DHCP_CLIENT_RUN.COM command procedure. TCPIP$DHCP_CLIENT_RUN invokes the DHCP client image, TCPIP$DHCP_CLIENT.EXE. TCPIP$DHCP_CLIENT continues to run until it is manually shutdown or the system is shutdown. Therefore, if more than one interface is to be configured, TCPIP$DHCP_CLIENT is ready to accept another DHCPCONF start command.

4. DHCP client accepts the DHCPCONF start command.

   DHCP client reads the start command and begins the DHCP dialog with the server. DHCPCONF and the DHCP client use a simple UDP-based protocol to communicate. If a HOSTNAME. file exists, the suggested host name is sent to the server.

5. The DHCP client/server DHCP dialog completes.

   DHCP client engages in a dialog with the DHCP server and when it completes the DHCP client sets the interface's IP address, subnet mask and broadcast address by sending the information via an `ioctl` to the TCP/IP kernel. If the interface is designated as the primary interface then any system-wide parameters received from the DHCP server are configured into the system.

6. DHCP client saves all parameters received from the DHCP server in a file (*interface*.DHC). This step occurs even if the interface is not designated as the primary interface.

7. DHCP client sends a task completion message to DHCPCONF to indicate that the interface is initialized and ready for work.

8. The START COMMUNICATION/INITIALIZE command then repeats this process for the next interface configured to be under DHCP control.

8.2 DHCP Client Components

The section describes the software and system elements that comprise DHCP client including:

- Executable files
- Configuration files
- Command files
- System logical names
- Log files

8.2.1 Executable Files

Three programs comprise the DHCP client component:

- TCPIP$DHCP_CLIENT.EXE

  This is the executable file for the DHCP agent or daemon. This process engages in the DHCP protocol dialog with the DHCP server, receives the parameters from the server and then configures the parameters on the local system. The parameters include IP addresses and their lease information, among others.

There is one DHCP client process per system, even for multihomed hosts. The DHCP client process is always running on a system that has an interface designated under DHCP control. The DHCP client uses the OpenVMS lock manager to prevent multiple DHCP client processes from executing concurrently on a system. The resource name used to control the number of client processes is TCPIP$DHCP_CLIENT_*scsnode*.

You stop this process by invoking the TCPIP$DHCP_CLIENT_SHUTDOWN command procedure or by sending a DHCPSIGTERM to the process using the TCPIP$DHCP_SIGNAL utility. Do not use the DCL command STOP/IDENTIFICATION to stop this process.

To ensure proper termination of the DHCP client process, Compaq recommends that you run the TCPIP$SHUTDOWN.COM procedure from your site-specific shutdown procedure.

When a DHCP client process is not already executing, and you or the system issues a DHCPCONF command, the system will automatically run the DHCP client process. The TCP/IP command SET INTERFACE/DHCP and the TCP/IP command START COMMUNICATION/INITIALIZE both invoke DHCPCONF to start the DHCP client. Running TCPIP$DHCP_CLIENT_STARTUP.COM will not itself create a DHCP client process.

- TCPIP$DHCP_CLIENT_CONF.EXE

  This is the executable file for the DHCPCONF command, which is the UNIX interface to the DHCP client. It communicates with the DHCP client process to start the client, release a lease, drop the interface from control and other requests.

  Most users do not need to execute a DHCPCONF command directly. The TCP/IP command SET INTERFACE/DHCP issues the necessary DHCPCONF commands.

- TCPIP$DHCP_CLIENT_SHOWDHC.EXE

  This is the executable file for the SHOWDHC command. This command displays the data stored in an interface's parameter file (*interface*.DHC). Refer to Section 8.6 for a description of the commands supported by this program.

8.2.2 Configuration Files

DHCP client uses the following files to control its environment:

- Configuration
- Interface
- Host name
- DHCPTAGS

8.2.2.1 Client Configuration File

DHCP client has one configuration file that controls DHCP client behavior. This optional file, named CLIENT.PCY, is an ASCII file located in the DHCP home directory, which is either SYS$SYSDEVICE:[TCPIP$DHCP] or a directory pointed to by the system logical TCPIP$DHCP_CONFIG. If CLIENT.PCY does not exist, DHCP client uses default values for each of the configurable parameters.

Example 8–1 shows the contents of a typical CLIENT.PCY file.

Example 8–1 Client Startup File

```
$ TYPE CLIENT.PCY
class_id TCPVMS
lease_desired 86400          #   24 hour lease
request routers
request host_name
request dns_servers
request dns_domain_name
```

The format of the configuration file must adhere to the following rules:

- Blank lines are ignored.
- The pound (#) character introduces a comment that continues to the next newline character.
- Each new policy option must begin and end on a separate line.
- Policy options are introduced by a keyword and may be Boolean, or they may take a value separated from the keyword by white space (but not a newline character).
- If an option is present more than once, only the value attached to the last occurrence will be take effect; earlier values are ignored.

Table 8–1 describes the configuration keywords.

Table 8–1 Configuration Keywords

Keyword	Description
`class_id`	Specifies the client's class identification. Consult RFC 2131 for details. The only class supported by TCP/IP Services is TCPVMS.
`lease_desired` *seconds*	Specifies that a client may request a lease of a particular duration, although DHCP servers are not bound to honor the request. If the client does not want a lease of a particular duration, *seconds* should be set to 0. If an infinite lease is required, set *seconds* to -1. Otherwise, specify (in seconds) the lease duration required. The default value is 0 seconds. A DHCP server grants a client permission to use an IP address for a fixed period of time, which may be infinite. In the language of DHCP, the client is granted a lease on the IP address.
`retries`	Specifies the maximum number of DHCPDISCOVER, DHCPOFFER, DHCP_REQUEST, DHCPNAK sequences the client attempts. An offer received and then refused is an unusual event; if it occurs more than once, this indicates a problem with the server. If you do not want to limit the number of bad offers that a client is willing to accept, set the value of this parameter to 0 (zero) or a negative value. The default value is 2 attempts.

(continued on next page)

Table 8–1 (Cont.) Configuration Keywords

Keyword	Description
start_delay *seconds*	Specifies the maximum time (in seconds) the client delays before broadcasting DHCP packets. When the DHCP client is invoked to configure an interface it will delay for a short time before broadcasting the first DHCP packet. The delay time is randomized from a value of 0 up to the value specified by *seconds*. The TCP/IP commands SET INTERFACE/DHCP and the START COMMUNICATION/INITIALZE command (executed at product startup time) both execute dhcpconf start commands and experience the randomized delay. The default value for *seconds* is 10 seconds.
timeouts *value,value,value....*	Specifies how long the client should wait for replies before timing out and retrying the broadcast. The DHCP protocol requires clients to implement an exponential retransmission and backoff when broadcasting discover or request packets. Each time the client sends a DHCP protocol packet, it waits for a response until a timeout occurs after an interval (in seconds) given by a member of the list of values. If a timeout occurs, the packet retransmits with the same XID (see RFC 1541), and the timeout is set to the next positive value in the comma-separated list. The last element in the list is negative or 0 (zero). At this point, the next action depends on options to the DHCPCONF program. One option is to fail. Another option is to retry forever. If the last value in the list is negative, DHCP suspends configuration of the interface for an amount of time given by the negative number terminating the list of values. During this time, the interface is considered idle—the client is not expecting responses destined for the interface and will ignore any that arrive. When the idle time is over, the client begins retransmitting with a new XID, and a timeout value is given by the first element in the array of values. If the last value is 0 (zero), the client continues to use the same XID and timeout of the last positive value in the list of values. The default list of values is 4, 8, 16, 32, 0.
use_saved_config	Specifies to use the configuration stored in *ifname*.DHC from a previous invocation of the protocol if the following conditions exist: • The lease is still valid. • There is no reply to DHCP. • use_saved_config is set.

(continued on next page)

Table 8–1 (Cont.) Configuration Keywords

Keyword	Description
`request` *`parameter_name`*	Specifies the parameter to request from the DHCP server. There may be many instances of the request keyword, each with a different *parameter_name*. Each parameter which is configurable through DHCP is identified by a unique parameter. Limited size of DHCP packets dictates that a client should not request data which it cannot use. Different implementations of DHCP servers or differing DHCP server policies can dictate that a server return more configuration parameters than a client requests. On the other hand, some DHCP servers will not send a parameter to a client unless the client explicitly requests it. If your DHCP server is configured to deliver a particular parameter to your TCP/IP Services DHCP client and the client is not receiving the information, verify that the DHCP client has a request statement for the information in its CLIENT.PCY file. Table 8–2 lists the DHCP parameters that a TCP/IP Services DHCP client may request from a server. Note that vendor-specific options, like the ones marked with TCPVMS in columns 3 and 4 of the DHCPTAGS. file entries, may not appear in a request statement.

Table 8–2 lists the Request statement parameters supported by the TCP/IP Services DHCP client implementation.

Table 8–2 Supported Request Parameters

Parameter Name	DHCP Option Code	This parameter requests...
Interface-specific parameters		
broadcast_address	28	The broadcast address in use on the client's subnet.
interface_mtu	26	The MTU size to use when performing Path MTU discovery.
subnet_mask	1	The client's subnet mask.
Systemwide parameters		
dns_domain_name	15	The domain name that the client should use when resolving host names using the Domain Name System (DNS).
dns_servers	6	A list of DNS name servers available to the client.
host_name	12	The host name of the client.
ip_time_to_live	23	The default time-to-live value the client should use on outgoing datagrams.
ip_forwarding	19	How the client should configure its IP layer for packet forwarding.
keepalive_interval	38	The time interval (in seconds) that the client TCP should wait before sending a keepalive message on a TCP connection.
routers	3	A list of IP addresses for routers on the client's subnet. Routers are listed in the order of preference.
static_routes	33	A list of static routes the client should install in its routing cache. If multiple routes to the same destination are specified, they are listed in descending order of priority. The routes consist of a list of IP address pairs. The first address is the destination address and the second address is the router for the destination.
tcp_default_time_to_live	37	The default time-to-live value that the client uses when sending TCP segments.

8.2.2.2 The Interface File

When the DHCP client receives parameters to configure the interface on the client, it stores them in a file named *ifname*.DHC along with the IP address lease information. The *ifname* part of the file name is the name of the interface on which the parameters were received. For example, the file created for parameters received on interface SE0 is SE0.DHC. There is one file per interface, and the files are placed in the directory specified by the system logical name TCPIP$DHCP_CONFIG (if it is defined) or in the SYS$SYSDEVICE:[TCPIP$DHCP] directory.

The interface file is a binary file, and you can display its contents by using the SHOWDHC utility. See Section 8.6 for information on how to use the SHOWDHC utility.

8.2.2.3 The Host Name File

You can configure the DHCP client to suggest a host name of your choice to the DHCP server by entering the name into a file named HOSTNAME.*ifname*. This file contains one line of text that contains the unqualified host name to suggest. You store the file in directory specified by the system logical TCPIP$DHCP_CONFIG, if defined, or in the SYS$SYSDEVICE:[TCPIP$DHCP] directory.

If you have multiple interfaces and want to suggest a different host name for each one, put the desired interface host names into separate files called HOSTNAME.*ifname*, where *ifname* is the name of the interface. For example, if you have two interfaces, WF0 and WE0, and you want the WF0 interface to receive the host name `myhostfiber` and the WE0 interface to receive the name `myhostether`, enter the following commands:

```
$ CREATE SYS$SYSDEVICE:[TCPIP$DHCP]HOSTNAME.WF0
  myhostfiber
  <CTRL-Z>
$ CREATE SYS$SYSDEVICE:[TCPIP$DHCP]HOSTNAME.WE0
  myhostether
  <CTRL-Z>
```

When configuring an interface, the DHCP client will first check for a HOSTNAME.*ifname* file and then, if that is not found, for the HOSTNAME. file.

When you initially configure the DHCP client the value of your node's SCSNODE parameter is placed into a file called HOSTNAME. with no *.ifname* extension.

If you change the HOSTNAME.*ifname* file, you must delete the *interface*.DHC file for the change to take effect.

8.2.2.4 The DHCPTAGS. File

The DHCPTAGS. file identifies the type of each parameter returned to the DHCP client by the DHCP server. Each supported option consists of the following:

- Option code number
- A two digit mnemonic code
- A short mnemonic text string for use in the DHCPCAP database
- A description of each option

The options are defined as follows:

- Standard

  The semantics on which all client and server DHCP implementations agree. These options are administered by the Internet Assigned Numbers Authority (IANA). They are numbered from 1 to 127, and 255.

- Site specific

  Within a specific site all client and server implementations agree on the semantics, but at another site the type and meaning of an option may differ. These options are numbered from 128 to 254.

- Vendor specific

  Each vendor may define 256 options unique to that vendor. The vendor is identified within a DHCP packet by the Vendor Class option (#60). An option with a specific numeric identifier belonging to one vendor will, in general, have a type and semantics different from those of another vendor. Vendor

options are super encapsulated into the vendor field (#43); within a specific DHCP packet there may be several instances of option #43.

- Pseudotags

  These are fields of the BOOTP packet and are not defined in RFC2131. Do not change these fields.

In general, the DHCP server knows little about the semantics of the first three options. Its only duty is to deliver those values to clients that need them. The responsibility for understanding and using the data rests with the client. Pseudotags have a meaning specific to TCP/IP Services.

8.2.3 Command Files

Table 8–3 lists the command files that the DHCP client uses to start up and shut down the component.

Table 8–3 DHCP Client Command Files

Command File Name	Description
TCPIP$DHCP_CLIENT_STARTUP.COM	Installs the DHCP client image.
TCPIP$DHCP_CLIENT_SHUTDOWN.COM	Stops DHCP client.

8.2.4 System Logicals

Use the logicals listed in Table 8–4 to alter the behavior of the DHCP client.

Table 8–4 DHCP Client System Logicals

Logical Name	Purpose
TCPIP$DHCP_DEBUG	Turns on DHCP client diagnostics. Refer to Section 7.2.4 for a description of this logical.
TCPIP$DHCP_CONFIG *directory*	Specifies the directory from which to read input files (CLIENT.PCY, DHCPTAGS. and HOSTNAME.) and to which to write output files (`ifname`.DHC). Note that DHCP client log files will still go to the default directory of the DHCP client account.
TCPIP$LOG_LEVEL *value*	Writes the specified level of diagnostic information to the log file. Ignored if TCPIP$DHCP_DEBUG is defined. Valid numeric values are: 0 No logging (default). 1 Log warning messages. 2 Log all messages.

8.2.5 Log Files

DHCP client creates a log file named TCPIP$DHCP_CLIENT_RUN.LOG in the directory SYS$SYSDEVICE:[TCPIP$DHCP].

8.3 DHCP Client Startup and Shutdown

The DHCP client can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$DHCP_CLIENT_STARTUP.COM allows you to start up the DHCP client service.
- SYS$STARTUP:TCPIP$DHCP_CLIENT_SHUTDOWN.COM allows you to shut down the DHCP client service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$DHCP_CLIENT_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when DHCP client is started.
- SYS$STARTUP:TCPIP$DHCP_CLIENT_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when DHCP client is shut down.

8.4 Configuring the DHCP Client

In order for the DHCP client to run, you must perform the following steps:

1. Put at least one interface under DHCP control.
2. Configure the DHCP client software.

8.4.1 Putting Interfaces under DHCP Control

For the DHCP client to execute, at least one interface on your host must be designated as being under DHCP control. This means that the interface IP address, subnet mask, and broadcast address are set automatically by DHCP when the system invokes the command procedure TCPIP$STARTUP.COM.

To place interfaces under DHCP control, you have these options:

- Use DHCP client autoconfigure for new TCP/IP Services installations.
- Use TCPIP$CONFIG to put interfaces under DHCP control.

8.4.1.1 Using Autoconfigure on a New TCP/IP Installation

If you have never installed a previous version of UCX or TCP/IP Services, you may simply install TCP/IP Services and manually invoke the SYS$STARTUP:TCPIP$STARTUP.COM procedure. TCPIP$STARTUP.COM detects that you have never run TCPIP$CONFIG and asks whether you want DHCP client to configure your host for you. If you answer Yes, TCPIP$STARTUP.COM invokes TCPIP$CONFIG to configure a small set of services and sets any unconfigured interfaces to be under DHCP control. This process is done in silent mode and asks you no questions.

The services enabled when you autoconfigure are:

- FTP client
- TELNET client
- TELNET server

If you want more than the set of services configured by this option, you can configure your host with the subset of TCP/IP Services and at a later time run TCPIP$CONFIG to configure other services.

DHCP client autoconfigure puts each unconfigured IP interface under DHCP control. It employs the following rules to decide which, if any, interface should be marked as the primary interface. (See Section 8.1.1 for an explanation of the DHCP primary interface.)

- If any interface currently has a permanent IP address, then TCPIP$CONFIG will not mark any of the interfaces under DHCP control as primary.
- If no interfaces are currently configured, then the first interface that TCPIP$CONFIG sees and marks as under DHCP control becomes the primary DHCP interface.

8.4.1.2 Using TCPIP$CONFIG to Configure an Existing Installation

If you have an existing TCP/IP installation, use TCPIP$CONFIG to place interfaces under DHCP control. To do this, perform the following steps:

1. From the TCPIP$CONFIG main menu, choose the Core Environment option and then choose the Interfaces option.
2. TCPIP$CONFIG presents a menu for each interface that it finds and gives you the option to:
   - Configure the interface manually.
   - Allow DHCP client to configure the interface.
   - Leave the interface unchanged.

   The following example illustrates this procedure:

```
INTERFACE Configuration

       The Ethernet device(s) on your system are: ESA0:

       Start of configuration questions for Internet interface SE0.

       SE0 is the Ethernet device ESA0:
 Interface: SE0
   IP_Addr: 10.0.0.1          NETWRK: 255.0.0.0         BRDCST: 10.255.255.255
    C_Addr:                 C_NETWRK:                 C_BRDCST:

   Flags:
   Receive buffer:            0

       Compaq TCP/IP Services for OpenVMS Interface SE0 Reconfiguration Menu

       Reconfiguration options:

                1  -  Configure interface manually       (Current default)
                2  -  Let DHCP configure interface

               [E] -  Exit menu (Do not reconfigure interface SE0)

Enter configuration option: 2

End of configuration questions for Internet interface SE0
```

3. If the system has multiple interfaces, DHCP client displays information about each existing interface and gives you the option to configure the interface manually or to allow DHCP configure the interface.

4. The next phase in the configuration process allows you to designate an interface as the primary DHCP interface.

```
        Primary DHCP Interface Configuration

         DHCP Client configures system-wide parameters and
         interface-specific parameters.  Only one interface, the
         DHCP "primary" interface, can receive system-wide
         parameters.

    Which interface? (SE0,NONE,HELP) [NONE]:SE0
```

5. At this point, TCPIP$CONFIG sets up the account for the DHCP client and default directory and initial copies of the required configuration and data files. For more information, see Section 8.4.2.

8.4.2 Configuring the Software

In order for the DHCP client to function, DHCP client software must be configured. As with any TCP/IP service, configuration involves:

- Creation of a default directory and an account in which the software can run
- Creation of data files

TCPIP$CONFIG.COM provides a menu option under the client menu called DHCP client. This option configures the DHCP client for you. You may choose this option explicitly, but if you put an interface under DHCP control from the Interfaces menu in TCPIP$CONFIG, this step is automatically done for you.

The DHCP client software configuration does the following:

- Creates the TCPIP$DHCP account, if it is not present.
- Creates the SYS$SYSDEVICE:[TCPIP$DHCP] directory, if it is not present.
- Enables the DHCP client service, if it is not present in the configuration database.

  Note that there is no service database entry for the DHCP client.
- Creates the initial versions of the following required configuration and data files.
  - DHCPTAGS.

    TCPIP$CONFIG extracts a copy of the DHCPTAGS. file from the librarian file, TCPIP$TEMPLATES.TLB. This file generally does not require modification. For a description of the DHCPTAGS. file, see Section 8.2.2.4.
  - CLIENT.PCY

    TCPIP$CONFIG extracts a copy of the CLIENT.PCY file TCPIP$TEMPLATES.TLB. This file governs the behavior of the DHCP client. Among other things, it tells the DHCP client which DHCP configurable parameters to request from the DHCP server. The file, as it comes from TCPIP$TEMPLATES.TLB, requests the most essential parameters from the server, including:

    Default route
    Host name
    DNS servers IP addresses
    DNS domain name

    See Section 8.2.2.1 for more information about this file.

– HOSTNAME.[*ifname*]

  This file contains a host name that you want to suggest that the DHCP server use as the system's host name. TCPIP$CONFIG puts the value of the cluster system parameter SCSNODE from the client system into this file. For more information about this file, see Section 8.2.2.3.

After extracting the files, TCPIP$CONFIG places the files into the directory pointed to by the TCPIP$DHCP_CONFIG logical, if it is defined. If TCPIP$DHCP_CONFIG is not defined, then the files are put into the SYS$SYSDEVICE:[TCPIP$DHCP] directory. No files are created if a version already exists.

Note

The DHCP client may not coexist on the same system as a DHCP server. TCPIP$CONFIG will not allow you to configure the DHCP client on a system with DHCP server configured.

8.4.3 Configuring a Cluster Environment

If you want to have multiple OpenVMS cluster nodes sharing the same CLIENT.PCY file and the nodes have identical interface names, a conflict will arise if you simply define TCPIP$DHCP_CONFIG to a common directory shared between the systems.

For example, if two systems in your cluster both have an interface named SE0 under DHCP control, to configure for this situation:

1. Define the system logical TCPIP$DHCP_CONFIG as a search list first pointing to a system-specific directory that you create for the DHCP client and then to the common directory.
2. Place the CLIENT.PCY file in the common directory.
3. If you want, place the HOSTNAME file into the SYS$SPECIFIC: directory. The *ifname*.DHC files will be created in the SYS$SPECIFIC:[ ] directory. For completeness, you might want to make the default device and directory for the TCPIP$DHCP account the SYS$SPECIFIC:[ ] directory, too.

8.4.4 Signaling the DHCP Client

You can use the TCPIP$DHCP_SIGNAL utility to signal the DHCP client to:

- Translate utility logicals and read configuration files
- Shutdown the DHCP client
- Dump the diagnostic state of the DHCP client to a file

Table 8–5 shows the commands available with the TCPIP$DHCP_SIGNAL utility:

Table 8–5 DHCP Signal Commands

Command	Description
DHCPSIGHUP	Causes the ASCII configuration files to be read again and then translates the TCPIP$DHCP_DEBUG and TCPIP$DHCP_LOG_LEVEL logicals.
DHCPSIGTERM	Causes an orderly shutdown of DHCP client. Use this command cautiously, as active lease and timer information is lost when you signal the DHCP client to shutdown. As a consequence, when you again start up the DHCP client, the system could be running with an expired lease.
DHCPSIGUSR1	Causes diagnostic state information to be written to the TCPIP$DHCP_CLIENT_RUN.LOG file.

8.5 TCP/IP Management Commands

You can use TCP/IP management commands to:

- Temporarily put an interface under DHCP control
- Permanently put an interface under DHCP control

8.5.1 Temporarily Configuring Interfaces

The TCP/IP command SET INTERFACE temporarily puts an interface under DHCP control. It does not make any change to the TCPIP$CONFIGURATION data base.

The format of the command is:

```
SET INTERFACE ifname/DHCP [/[NO]PRIMARY]
```

In this format, *ifname* is the name of the interface; for example, SE0.

You must enter the SET NOINTERFACE command on the interface before entering a SET INTERFACE/DHCP.

After you enter this command, the interface receives a new IP address from the DHCP server, but the information stored in the TCPIP$CONFIGURATION.DAT file is unchanged. For example, if you issue the TCP/IP command SHOW CONFIGURATION INTERFACE for the interface you see the IP address you had set up for the interface before you temporarily configured the interface. In addition, when you stop and restart TCP/IP Services, the interface will have the previously assigned IP address.

If you want the interface to be permanently under DHCP control, you must either run the TCPIP$CONFIG.COM command procedure to put the interface under DHCP control or enter the SET CONFIGURATION INTERFACE *ifname*/DHCP command. Compaq recommends that you run the TCPIP$CONFIG.COM command procedure.

8.5.2 Permanently Configuring Interfaces

The TCP/IP command SET CONFIGURATION INTERFACE /DHCP configures an interface to be under DHCP control by adding or changing an entry in the TCPIP$CONFIGURATION database. After entering this command, every time TCPIP$STARTUP.COM is run the DHCP client is invoked to configure the interface.

Note that this command does not change the current run-time configuration of the interface. For any changes to the TCPIP$CONFIGURATION database to take effect, you must run $TCPIP$STARTUP or enter a TCP/IP command START COMMUNICATION/INITIALIZE.

The format of the command is:

```
SET CONFIGURATION INTERFACE ifname/DHCP [/[NO]PRIMARY]
```

In this format, *ifname* is the name of the interface; for example, SE0.

The optional qualifier /PRIMARY indicates that the interface is to be the primary DHCP client interface. (See Section 8.1.1 for a description of the DHCP client primary interface.) TCP/IP Services issues an error if one of the other interfaces has the primary designation.

/NOPRIMARY indicates that the interface is no longer to be marked as the primary DHCP client interface. It is not an error if turning off this option leaves no primary DHCP interfaces, because a primary DHCP interface is not required.

You do not need to issue this command because TCPIP$CONFIG executes the command for you when you put an interface under DHCP control.

8.6 Using the SHOWDHC Utility

TCP/IP Services provides the SHOWDHC utility for displaying the contents of an interface parameter file.

The SHOWDHC utility displays data stored in an *ifname*.DHC file.

The format of the SHOWDHC utility command is as follows:

```
SHOWDHC filename
```

In this format, *filename* is the name of an *ifname*.DHC file.

The format of the SHOWDHC output is a single line in the format of the DHCPCAP. file. For more information on the format of the DHCPCAP. file, refer to Section 7.2.2.2. Example 8–2 shows typical output from the SHOWDHC utility.

Example 8–2 SHOWDHC Sample Output

```
$ SHOWDHC SE0.DHC
se0.dhc:
ht=1:ha=08.00.2b.2a.de.a8:sa=10.10.2.3:yi=10.10.2.101:sm=255.255.255.0:gw=10.10.
2.66:ds=10.10.2.11:ho=rufus:dn=lkg.dec.com:ba=10.10.2.255:lt=1200:sv=10.10.2.3:
t1=600:t2=1050:
```

9
Configuring BOOTP

The Bootstrap Protocol (BOOTP) server answers network bootstrap requests from diskless workstations and other network devices such as routers, terminal servers, and network switching equipment. When it receives such a request, the BOOTP server looks up the client's address in the BOOTP database file.

The Trivial File Transfer Protocol (TFTP) handles the file transfer from a TFTP server to a diskless client or other remote system. The client initiates the file transfer. TFTP is described in Chapter 10.

Because BOOTP is a subset of DHCP, you cannot enable both BOOTP and DHCP on the same host.

This chapter reviews key concepts and describes:

- How to plan for configuring BOOTP (Section 9.2)
- How to configure the BOOTP service (Section 9.3)
- How to manage the BOOTP service (Section 9.4)
- Create the BOOTP database and populate it with client entries (Section 9.5)
- Solve BOOTP problems (Section 9.6)

9.1 Key Concepts

The BOOTP server answers client requests for diskless client configuration by sending address and file name information to the client. When the client receives this information from the BOOTP server, it initiates a file transfer using the TFTP protocol.

Performing a network bootstrap consists of the following steps for the BOOTP server:

1. The BOOTP server receives a configuration request from a client. A broadcast request goes out to all potential servers on the subnetwork or is directed to a predetermined known server address.
2. The BOOTP server reads information in the BOOTP database to get information about the client. The identity of the client is based on the network hardware address contained in the request.
3. BOOTP identifies the network client.
4. BOOTP constructs a response that contains all of the information in the BOOTP database for that client. The client information in the database includes:
   - Client's IP address
   - Client's host name (usually)

- Name and size of the client's system load file
- IP address of the TFTP server storing this file
- IP addresses of the hosts offering common network services, such as a log server or a print (LPD) server.

5. When the client receives the configuration information in the BOOTP response, it sends a request to the TFTP server host named in the response. This request is necessary only if the client must retrieve the load file.
6. If the client sends a read request (RRQ) to the TFTP server, it attempts to locate this file. If it finds the file, the server transfers it to the client.

9.2 BOOTP Planning and Preconfiguration Tasks

When planning BOOTP, you need to make decisions about the network configuration and the local BOOTP service.

9.2.1 Network Configuration Decisions

Before you start to set up BOOTP, answer the following questions:

- What clients will access the BOOTP server? For each client, obtain the following information:
  - System image and location from where it can be copied
  - Additional information requested
  - Hardware address
  - IP address
- What hosts in your network will run the BOOTP server?
- Will gateways be used for downloading? Gateways let you specify a specific path for the data transfer.
- Do you want to limit client access to specific server directories?

9.2.2 BOOTP Service Decisions

Before you start to configure BOOTP, consider the following:

- Default priority for the TCPIP$BOOTP server account in the user authorization file (UAF)

  For optimal performance, use the default priority level for the TCPIP$BOOTP user account.

  In a large or active subnetwork, clients might generate several broadcast requests per minute. The server continues to process all incoming requests, even those for which it lacks information in its database.

  In most cases, all this processing does not create system performance problems. However, it does use, perhaps unnecessarily, system resources. A different network configuration might avoid wasted system overhead.

- Segmented subnetworks

  To reduce large volumes of BOOTP request traffic to a specific server, segment very large subnetworks with filtering bridges.

If you configure multiple servers, each server competes to provide the requested configuration information. For efficient use of each server, partition the database with a subset of the overall client population designated to each server.

- Separate directory for each client

  To avoid writing over the same file name with configuration information from multiple clients, create a separate subdirectory for each client in the TCPIP$TFTP_ROOT directory tree.

  Some BOOTP clients, such as routers and terminal servers, can store configuration options on the BOOTP server host. In a network with two or more of these clients, the clients can use the same file name to store the configuration information with TFTP.

- Security needs

  Identify your system's security needs (see Section 9.2.3).

9.2.3 BOOTP Security

For security purposes, the server runs as an unprivileged image that can access only the directories and files for which it has read access.

Compaq recommends that you safeguard your system's normal file protection mechanisms from unauthorized access. In particular, ensure the security of system files.

The BOOTP server runs as the nonprivileged OpenVMS user account TCPIP$BOOTP. When you set up BOOTP, follow these security procedures:

- Ensure that neither server has automatic access to any files.

  To make files accessible to the BOOTP server, grant appropriate access to its account. Use the normal OpenVMS file protection procedures. Enter the DCL command DIRECTORY/SECURITY to display the current file protection settings on a directory.

- Prevent unauthorized access to sensitive system or user data. Before you enable BOOTP, ensure that you have set up all the necessary file protections.
- Give the TCPIP$BOOTP user account read access to the files in the TCPIP$TFTP_ROOT: directory tree that might be used for downloading.
- Some clients first send a BOOTP request for the name of the file that they need downloaded. On receipt, BOOTP opens the file for read access and retrieves its size. BOOTP needs access to confirm that the file exists and to provide the size of the file to the client in the BOOTP response.

  Ensure that BOOTP has access to this file.

9.3 Configuring the BOOTP Service

To set up the BOOTP server software, run TCPIP$CONFIG (see the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual).

The procedure creates:

- BOOTP user account
- Service records in the services database
- Default directories

- Empty TCPIP$BOOTP database file

9.4 Managing the BOOTP Service

The following sections describe how to manage the BOOTP service.

9.4.1 Enabling and Disabling BOOTP

To enable and disable BOOTP, use these commands:

- On the running system:
  - ENABLE SERVICE BOOTP
  - DISABLE SERVICE BOOTP
- In the configuration database:
  - SET CONFIGURATION ENABLE SERVICE BOOTP
  - SET CONFIGURATION DISABLE SERVICE BOOTP

To check whether these services are enabled or disabled, enter the following commands:

- SHOW SERVICE BOOTP
- SHOW CONFIGURATION ENABLE SERVICE BOOTP

The following examples show how to use the SHOW SERVICE command to get information about BOOTP.

1. To display information about the BOOTP server processes, enter the SHOW SERVICE command. For example:

```
TCPIP> SHOW SERVICE BOOTP

Service         Port  Proto   Process       Address    State
BOOTP             67  UDP     TCPIP$BOOTP   0.0.0.0    Enabled
```

2. To display BOOTP service settings and statistics, include the /FULL qualifier. For example:

```
TCPIP> SHOW SERVICE BOOTP /FULL

Service: BOOTP
                    State:     Enabled
Port:        67     Protocol:  UDP             Address:  0.0.0.0
Inactivity:   5     User_name: TCPIP$BOOTP     Process:  TCPIP$BOOTP
Limit:        1     Active:      1             Peak:       1

File:  TCPIP$SYSTEM:TCPIP$BOOTP_RUN.COM
Flags: Listen

Socket Opts:  Rcheck Scheck
 Receive:            0     Send:               0

Log Opts:     Acpt Actv Dactv Conn Error Exit Logi Logo Mdfy Rjct TimeO Addr
 File:        SYS$SYSDEVICE:[TCPIP$BOOTP]TCPIP$BOOTP_RUN.LOG

Security
 Reject msg:  not defined
 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0
```

9.4.2 BOOTP Management Commands

Table 9–1 summarizes the BOOTP management commands.

Table 9–1 BOOTP Management Commands

Command	Function
CONVERT/VMS BOOTP	Populates an existing BOOTP database with entries from a UNIX /etc/bootptab file.
CREATE BOOTP	Creates an empty BOOTP database.
SET BOOTP	Adds or modifies client entries to the BOOTP database.
SHOW BOOTP	Displays client information from the BOOTP database.
ENABLE SERVICE BOOTP	Dynamically enables the BOOTP service.
DISABLE SERVICE BOOTP	Dynamically disables the BOOTP service.
SET CONFIGURATION ENABLE SERVICE BOOTP	Sets the configuration database to enable BOOTP at product startup.
SET CONFIGURATION DISABLE SERVICE BOOTP	Sets the configuration database to disable BOOTP at product startup.
SET SERVICE BOOTP	Configures the BOOTP service in the services database.
SET NOSERVICE BOOTP	Disables the BOOTP service in the configuration database.
SHOW SERVICE BOOTP	Displays BOOTP server information stored in the services database.

9.4.3 BOOTP Logical Names

Table 9–2 lists the logical names you can use to manage the BOOTP software.

Table 9–2 BOOTP and TFTP Logical Names

Name	Function
TCPIP$BOOTP	Points to the location of the BOOTP database file.
TCPIP$TFTP_ROOT	Defines a concealed device. Points to the TFTP data storage tree, for example, SYS$SYSDEVICE:[TCPIP$TFTP_ROOT.].
TCPIP$BOOTP_TRACE	Displays the client hardware address for every incoming BOOTP request and response to requests.

9.4.4 BOOTP Startup and Shutdown

The BOOTP service can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted. The following files are provided:

- SYS$STARTUP:TCPIP$BOOTP_STARTUP.COM allows you to start up BOOTP.
- SYS$STARTUP:TCPIP$BOOTP_SHUTDOWN.COM allows you to shut down BOOTP.

To preserve site-specific parameter settings and commands, you can create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$BOOTP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when BOOTP is started.
- SYS$STARTUP:TCPIP$BOOTP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when BOOTP is shut down.

9.5 Creating a BOOTP Database

If you choose to configure BOOTP while configuring TCP/IP Services, TCPIP$CONFIG creates an empty BOOTP database.

If you need to create it manually, use the TCP/IP management command CREATE BOOTP. This command creates the file SYS$SYSTEM:TCPIP$BOOTP.DAT. The command uses the logical name TCPIP$BOOTP to point to the BOOTP database file. To create a separate database, perhaps in a different disk directory or with a different file name, modify this logical name.

To create a temporary, separate, and empty BOOTP file, you can use a process-specific logical name. However, Compaq does not recommend creating separate or private BOOTP databases because the TCPIP$BOOTP user account requires read access to the database file.

9.5.1 Populating the BOOTP Database

For each BOOTP client in the BOOTP database, use the SET BOOTP command to enter the following required information:

- Client's hardware address (required).
- Either the client's name or IP address (required).
- Network mask (required).
- Client's system image file name (required).
- Interim gateway (routing) systems.
- Either the name or IP address of other network servers. Some of the optional servers that you can specify are:
  - Cookie servers
  - IEN-116 name servers
  - IMPRESS network image servers
  - LPR print servers
  - MIT-LCS UDP logging servers
  - DNS (BIND) name servers
  - Resource location (RLP) servers
  - Network time servers

To populate the BOOTP database with client entries, use these commands:

- CONVERT/VMS BOOTP (adds UNIX client records)
- SET BOOTP (adds individual client records)

9.5.2 Converting UNIX Records

You can use the BOOTP client information in an existing UNIX boot file. The CONVERT/VMS BOOTP command populates the existing BOOTP database with entries from a BIND formatted UNIX /etc/bootptab file.

Before you enter CONVERT/VMS BOOTP, define the logical name TCPIP$BOOTP. The CONVERT/VMS BOOTP command uses it to determine the directory and file name for the database. Enter the following command:

```
$ DEFINE /SYSTEM TCPIP$BOOTP SYS$COMMON:[SYSEXE]TCPIP$BOOTP.DAT
```

If you do not define TCPIP$BOOTP, the database is created as [*current_directory*]TCPIP$BOOTP.DAT.

To populate the BOOTP database by using entries in a UNIX /etc/bootptab file, follow these steps:

1. Copy the /etc/bootptab file to your system.
2. Edit the output file. Examine the directory path for each client entry. Modify the UNIX path names to OpenVMS specifications. For example, change:

   ```
   :hd=/usr/apple/orange/bootptab:
   ```

   to

   ```
   :hd="DISK_BIRD2$:[USR.APPLE.ORANGE]BOOTPTAB.DAT":
   ```

   Note that this is a UNIX file and is not compatible with OpenVMS.
3. Enter the CONVERT command as follows:

   ```
   TCPIP> CONVERT /VMS BOOTP
   ```

   The command reads the entries in your edited output file and adds them to the BOOTP database. If it finds an existing record for a client with a converted record, and if the information differs, the command updates the existing record with the newer data.

The CONVERT/VMS BOOTP command has the following format:

```
CONVERT/VMS BOOTP source_file /ADD_HOST /FILE=sys_image_file
```

In this command format:

- *source_file*

  Specifies the name of the file you edited (the output from the COPY command). The default is ETC.BOOTPTAB.

- /ADD_HOST

  Adds client entries that are new to your system to the hosts database. The default is not to add client entries to the hosts database.

- /FILE=*sys_image_file*

  Specifies the download file. Use this parameter if you are adding new clients to the BOOTP database. All these new clients have the same download file.

9.5.3 Creating Individual Entries

To add individual entries to the BOOTP database, enter:

```
TCPIP> SET BOOTP host /FILE=download_file/HARDWARE=ADDRESS=hex_address
```

In the following example, the SET BOOTP command adds host PLOVER, with hardware address 08-00-2D-20-23-21, to the BOOTP database. Note that the SET BOOTP command accepts as a parameter either the host name or the host's IP address. In the following example, the host name is specified:

```
TCPIP> SET BOOTP PLOVER /HARDWARE=ADDRESS=08-00-2D-20-23-21 /FILE=PLOVER.SYS
```

To display the BOOTP database, enter the SHOW BOOTP command, as follows:

```
TCPIP> SHOW BOOTP

Host                                    Hardware address

10.10.2.3                               08-00-00-20-23-21
10.10.2.120                             08-00-2B-A2-20-49
10.10.2.22                              08-00-2D-20-23-21
```

9.5.4 Modifying and Deleting Entries

To modify a record in the BOOTP database, use the SET BOOTP command. For example, to stop using hosts `seagull`, `tern`, and `sandpiper` as gateways for downline loading to PLOVER, enter:

```
TCPIP> SET BOOTP PLOVER /NOGATEWAYS=(seagull,tern,sandpiper)
```

To delete an entry from the BOOTP database, use the SET NOBOOTP command.

9.6 Solving BOOTP Problems

Most problems with BOOTP are due to:

- Inaccurate client information in the BOOTP database.
- Directory access restrictions because the TCPIP$BOOTP user account is not privileged.
- File access restrictions because the TCPIP$BOOTP user account is not privileged.

If BOOTP fails to respond to a client request, follow these steps:

1. Verify the accuracy of the information in the BOOTP database for that client, especially the hardware address and image file name.
2. Turn on logging.
3. Ensure that the BOOTP server has access to directories and files.
4. Set directory and file protections appropriately.

The BOOTP server ignores incoming requests from unknown clients (for example, clients that are not found in the BOOTP database). Therefore, it can be difficult to identify why incoming requests are not serviced.

By default, BOOTP does not generate logging information, even though it opens the file SYS$SYSDEVICE:[TCPIP$BOOTP]TCPIP$BOOTP_RUN.LOG. If you turn on logging, the log displays the client hardware address for every incoming BOOTP request, as well as any information used in response to those requests. With this information, you can detect whether the server sees a particular client

request. To turn on logging, define the following logical name. To activate the logical, shut down and restart the BOOTP service. For example:

```
$ DEFINE /SYSTEM TCPIP$BOOTP_TRACE 1
$ @SYS$STARTUP:TCPIP$BOOTP_SHUTDOWN.COM
$ @SYS$STARTUP:TCPIP$BOOTP_STARTUP.COM
```

Remove the logical names and restart BOOTP as soon as the problem is fixed. On a busy network with frequent BOOTP requests, the log file can rapidly consume large amounts of space on your system disk.

10
Configuring TFTP

The Trivial File Transfer Protocol (TFTP) handles the file transfer from a TFTP server to a diskless client or other remote system. The client initiates the file transfer.

The Bootstrap Protocol (BOOTP) server answers network bootstrap requests from diskless workstations and other network devices such as routers, terminal servers, and network switching equipment. For more information about setting up the BOOTP service, see Chapter 9.

This chapter reviews key concepts and describes:

- How to set up the TFTP service (Section 10.2)
- TFTP security (Section 10.3)
- How to solve TFTP problems (Section 10.4)

10.1 Key Concepts

The Trivial File Transfer Protocol (TFTP) transfers files from a TFTP server to diskless clients or other remote systems. The client initiates the file transfer.

If the client sends a read request to the TFTP server, the server attempts to locate this file.

TFTP has the following characteristics:

- TFTP clients are not registered in a database.
- TFTP, which runs as an unprivileged user in the TCPIP$TFTP account, is restricted to those files that the normal unprivileged user can access.
- TFTP clients are not regulated by the usual OpenVMS user security methods.
- No user name or password is required to use the TFTP service.

10.2 Setting up the TFTP Service

To set up the TFTP server software, run TCPIP$CONFIG (see the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual).

The procedure creates:

- A TFTP user account
- Service records in the services database
- Default directories
- A TFTP root directory to which the logical name TCPIP$TFTP_ROOT: will point

10.2.1 Transferring Data to the TFTP Host

The TFTP server allows clients to transfer data and program images to the TFTP server host. However, before the data transfer, a file must be created on the TFTP server host to which the data is transferred. This process controls the creation of files on the host, preventing unwanted files from being created on the TFTP host.

Each incoming transfer of data to a file creates a new version of the target file. As a result, you must manage the consumption of disk space on the server system by carefully setting up file version limits for the target files and directories.

To limit the number of versions of a file that can be created in a new directory, include the /VERSION_LIMIT qualifier on the DCL command CREATE/DIRECTORY. For example:

```
$ CREATE/DIRECTORY/VERSION_LIMIT=10 [MYPROJECT.SAVE]
```

For more information about managing the directories and files for TFTP transfers, see Section 10.3.

10.2.2 TFTP Management Commands

Table 10–1 summarizes the TFTP management commands.

Table 10–1 TFTP Management Commands

Command	Function
ENABLE SERVICE TFTP	Enables the TFTP service.
DISABLE SERVICE TFTP	Disables the TFTP service.
SET SERVICE TFTP	Configures TFTP in the service database.
SET NOSERVICE TFTP	Disables TFTP in the service database.
SHOW SERVICE TFTP	Displays information about TFTP from the service database.

10.2.3 TFTP Logical Names

The logical name described in Table 10–2 can be used to modify the behavior of the TFTP service:

Table 10–2 TFTP Logical Names

Name	Function
TCPIP$TFTP_ROOT	Defines a concealed device that points to TFTP data storage. By default, the concealed device is SYS$SYSDEVICE:[TCPIP$TFTP_ROOT]. For more information, see Section 10.3.

10.2.4 TFTP Startup and Shutdown

The TFTP service can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted. The following files are provided:

- SYS$STARTUP:TCPIP$TFTP_STARTUP.COM allows you to start up TFTP separately.

- SYS$STARTUP:TCPIP$TFTP_SHUTDOWN.COM allows you to shut down TFTP separately.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$TFTP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when TFPT is started.
- SYS$STARTUP:TCPIP$TFTP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when TFTP is shut down.

10.2.5 Enabling and Disabling TFTP

To enable and disable TFTP, use these commands:

- On the running system:
  - ENABLE SERVICE TFTP
  - DISABLE SERVICE TFTP
- In the configuration database:
  - SET CONFIGURATION ENABLE SERVICE TFTP
  - SET CONFIGURATION DISABLE SERVICE TFTP

To check whether these services are enabled or disabled, enter these commands:

- SHOW SERVICE TFTP
- SHOW CONFIGURATION ENABLE SERVICE TFTP

The following command shows how to obtain complete information about TFTP settings and statistics:

```
TCPIP> SHOW SERVICE TFTP /FULL

Service: TFTP
                          State:     Enabled
Port:            69     Protocol:  UDP            Address:  0.0.0.0
Inactivity:       5     User_name: TCPIP$TFTP     Process:  TCPIP$TFTP
Limit:            1     Active:         1         Peak:           1

File:         SYS$SYSDEVICE:[TCPIP$TFTP]TCPIP$TFTP_RUN.COM
Flags:        Listen

Socket Opts:  Rcheck Scheck
 Receive:            0     Send:               0

Log Opts:     Acpt Actv Dactv Conn Error Exit Logi Logo Mdfy Rjct TimO Addr
 File:        SYS$SYSDEVICE:[TCPIP$TFTP]TCPIP$TFTPD_RUN.LOG

Security
 Reject msg:  not defined
 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0
```

10.3 TFTP Security

For security purposes, the server runs as an unprivileged image that can access only the directories and files for which it has read access.

Compaq recommends that you safeguard your system's normal file protection mechanisms from unauthorized TFTP access. In particular, ensure the security of system files.

A client's download request can use one of several formats for its file name specification:

- If the unprivileged TFTP server has read access to the requested file, the client uses a fully qualified file name, including the device, directory, name, and extension, to directly access the file.
- If the client specifies only the file name and extension, the TFTP server attempts to locate the file in the default TFTP directory tree.

  You can designate this directory tree with the system logical name TCPIP$TFTP_ROOT:. This is a concealed device name, usually pointing to the directory SYS$SYSDEVICE:[TCPIP$TFTP_ROOT]. When looking for a directory, the TFTP server looks first in the TCPIP$TFTP_ROOT: area with the same name as the requesting client's host name.

For example, if a client named GULL.SHORE.COM sends a read request for the file SERVICE.DAT, the server's first attempt to find the file is in TCPIP$TFTP_ROOT:[GULL]. If that directory does not exist, the server next looks in the TCPIP$TFTP_ROOT: root directory, for example, in TCPIP$TFTP_ROOT:[000000]SERVICE.DAT.

If the TFTP client requests a file by specifying a name in UNIX format (for example, `/etc/gull/myfile`), TFTP translates this file specification into OpenVMS format.

The TFTP server runs as the nonprivileged OpenVMS user accounts TCPIP$TFTP. When you set up TFTP, follow these security procedures:

- Ensure that neither server has automatic access to any files.

  To make files accessible to the TFTP server, grant appropriate access to its account. Use the normal OpenVMS file protection procedures. For example, enter the DCL command DIRECTORY/SECURITY.
- Prevent unauthorized access to sensitive system or user data. Before you enable TFTP, ensure that you have set up all the necessary file protections.
- Give the TCPIP$TFTP user account read access to the files in the TCPIP$TFTP_ROOT: directory tree that might be used for downloading.

10.4 Solving TFTP Problems

The TFTP server is restricted to accessing only files or directories that OpenVMS file system security measures allow. Verify that these files have the appropriate protection and ownership so that the TFTP server has access to them. See Section 10.3 for more information.

- Ensure that the TFTP server has access to directories and files. Set protections accordingly.
- Create the target files to enable TFTP to reply to write requests.

The log file, SYS$SYSDEVICE:[TCPIP$TFTP]TCPIP$TFTP_RUN.LOG, can be useful for troubleshooting TFTP transfer failures.

11

Configuring the Portmapper

The Portmapper service eliminates the need to preconfigure all client and server remote procedure call (RPC) applications with the port numbers they use. The Portmapper "listens" at port 111 and maintains a database of registered server programs, their unique program numbers, and assigned port numbers.

This chapter describes:

- How to configure the services that use RPC with information that the Portmapper needs (Section 11.1)
- How to start up and shut down the Portmapper (Section 11.2)
- How to display Portmapper settings (Section 11.3)

For information about programming with the RPC application programming interface (API), see the *DIGITAL TCP/IP Services for OpenVMS ONC RPC Programming* manual.

11.1 Configuring Services to Use the Portmapper

You must run the Portmapper if you intend to use the following applications:

- MOUNT
- NFS Server
- PC-NFS
- Any customer-developed programs that use RPC

When you configure these services with TCPIP$CONFIG, you will be automatically prompted to set up the Portmapper service. The Portmapper service is then started when you start TCP/IP Services.

The SET SERVICE command configures the applications so they are known to the Portmapper. To set RPC-related parameters, use the /RPC qualifier. Enter:

```
TCPIP> SET SERVICE service -
_TCPIP> /RPC=(PROGRAM_NUMBER=n, VERSION_NUMBER=(LOWEST=n, HIGHEST=n))
```

The TCPIP services that use the Portmapper have the following default values for the /RPC qualifier:

Service	Default Program Number	Default Lowest Version	Default Highest Version
MOUNT	100005	1	3
NFS Server	100003	2	3

Service	Default Program Number	Default Lowest Version	Default Highest Version
PC-NFS	150001	1	2
PORTMAPPER	100000	1	1

11.2 Portmapper Startup and Shutdown

The Portmapper service can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$PORTMAPPER_STARTUP.COM allows you start up the Portmapper service separately.
- SYS$STARTUP:TCPIP$PORTMAPPER_SHUTDOWN.COM allows you to shut down the Portmapper service separately.

To preserve site-specific parameter settings and commands, you can create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$PORTMAPPER_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters used in the Portmapper startup procedure.
- SYS$STARTUP:TCPIP$PORTMAPPER_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters used in the Portmapper shutdown procedure.

11.3 Displaying Portmapper Information

The following examples show a variety of commands you can use to get information about the Portmapper and the services that depend on it.

1. The following example displays the RPC options for these running services: MOUNT, NFS, PC-NFS, and the Portmapper.

```
TCPIP> SHOW SERVICE /RPC /PERMANENT

                          RPC              Protocol Versions
Service             Program Number          Lowest / Highest

MOUNT                    100005                 1        3
NFS                      100003                 2        3
PCNFS                    150001                 1        2
PORTMAPPER               100000                 2        2
TCPIP>
```

2. In the following example, the /FULL and /PERMANENT qualifiers display the RPC options for the NFS server, whose program number is 100003, lowest version is 2, and highest version is 3.

```
TCPIP> SHOW SERVICE NFS /FULL /PERMANENT

Service: NFS

Port:             2049     Protocol:  UDP            Address:  0.0.0.0
Inactivity:          0     User_name: TCPIP$NFS        Process:  TCPIP$NFS
Limit:               1
```

```
File:         TCPIP$SYSTEM:TCPIP$NFS_RUN.COM
Flags:        TCPIP

Socket Opts:  Rcheck Scheck
 Receive:        64000     Send:           64000

Log Opts:     Acpt Actv Dactv Conn Error Exit Logi Logo Mdfy Rjct TimO Addr
 File:        SYS$SYSDEVICE:[TCPIP$NFS]TCPIP$NFS_RUN.LOG

RPC Opts
 Program number:      100003  Low version:      2   High version:      3

Security
 Reject msg:  not defined
 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0
TCPIP>
```

3. The following example shows how to display information about all the registered applications:

```
TCPIP> SHOW PORTMAPPER

    Program Number     Version  Protocol  Port-number  Process   Service-name
--------------------  -------  --------  -----------  --------  ------------
000186A0 (    100000)        2  TCP          111        00000060  PORTMAPPER
000186A0 (    100000)        2  UDP          111        00000060  PORTMAPPER
000186A5 (    100005)        1  UDP           10        00000064  MOUNT
000186A5 (    100005)        3  UDP           10        00000064  MOUNT
000186A5 (    100005)        1  TCP           10        00000064  MOUNT
000186A5 (    100005)        3  TCP           10        00000064  MOUNT
000186A3 (    100003)        2  TCP         2049        00000065  NFS
000186A3 (    100003)        2  UDP         2049        00000065  NFS
000186A3 (    100003)        3  TCP         2049        00000065  NFS
000186A3 (    100003)        3  UDP         2049        00000065  NFS
```

4. The following example shows how to monitor the server:

```
TCPIP> SHOW SERVICE PORTMAPPER

Service     Port      Protocol        Process     Address     State

PORTMAPPER   111       TCP,UDP      TCPIP$PORTM     0.0.0.0     Enabled

TCPIP>
```

12

Configuring and Managing NTP

The Network Time Protocol (NTP) synchronizes time and coordinates time distribution throughout a TCP/IP network. NTP provides accurate and dependable timekeeping for hosts on TCP/IP networks. TCP/IP Services NTP software is an implementation of the NTP Version 3 specification and maintains compatibility with NTP versions 1 and 2.

NTP provides synchronization traceable to clocks of high absolute accuracy and avoids synchronization to clocks keeping incorrect time.

Time synchronization is important in client/server computing. For example, systems that share common databases require coordinated transaction processing and timestamping of instrumental data.

This chapter reviews key concepts and describes:

- How to start up and shut down NTP (Section 12.2)
- How to configure the NTP host (Section 12.3)
- How to configure the host as a backup time server (Section 12.4)
- How to adjust time zone offsets (Section 12.5)
- NTP event logging (Section 12.6)
- How to configure NTP authentication (Section 12.7)
- How to use NTP utilities (Section 12.8)
- How to solve NTP problems (Section 12.9)

12.1 Key Concepts

Synchronized timekeeping means that hosts with accurate system timestamps send time quotes to each other. Hosts running NTP may be either time servers or clients although they are often both servers and clients.

NTP does not attempt to synchronize clocks to each other. Rather, each server attempts to synchronize to Universal Coordinated Time (UTC) using the best available source and best available transmission paths to that source. NTP expects that the time being distributed from the root of the synchronization subnet will be derived from some external source of UTC (for example, a radio clock).

If your network is isolated and you cannot access other NTP servers on the internet, you can designate one of your nodes as the reference clock to which all other hosts will synchronize.

12.1.1 Time Distributed Through a Hierarchy of Servers

In the NTP environment, time is distributed through a hierarchy of NTP time servers. Each server adopts a stratum that indicates how far away it is operating from an external source of UTC. NTP times are an offset of UTC. Stratum 1 servers have access to an external time source, usually a radio clock. A stratum 2 server is one that is currently obtaining time from a stratum 1 server; a stratum 3 server gets its time from a stratum 2 server, and so on. To avoid long-lived synchronization loops, the number of strata is limited to 15.

Stratum 2 (and higher) hosts might be company or campus servers that obtain time from some number of primary servers and provide time to many local clients. In general:

- Lower-strata hosts act as time servers.
- Higher-strata hosts are clients that adjust their time clocks according to the servers.

Internet time servers are usually stratum 1 servers. Other hosts connected to an internet time server have stratum numbers of 2 or higher and may act as time servers for other hosts on the network. Clients usually choose one of the lowest accessible stratum servers from which to synchronize.

12.1.2 How Hosts Negotiate Synchronization

Each host has its identifying stratum number encoded within UDP datagrams. Peers communicate by exchanging these timestamped UDP datagrams. NTP uses these exchanges to construct a list of possible synchronization sources then sorts them according to stratum and synchronization distance. Peers are accepted or rejected, leaving only the most accurate and precise sources.

NTP evaluates any new peer to determine whether it qualifies as a new (more suitable) synchronization source.

NTP rejects the peer under the following conditions:

- The peer is not synchronized.
- The stratum is higher than the current source's stratum.
- The peer is synchronized to the local node.

NTP accepts the peer under the following conditions:

- There is no current time source.
- The current source is unreachable.
- The current source is not synchronized
- The new source's stratum is lower than the current source.
- The new source's stratum is the same as the current source, but its distance is closer to the synchronization source by more than 50%.

12.1.3 How the OpenVMS System Maintains the System Clock

The OpenVMS system clock is maintained as a software timer with a resolution of 100 nanoseconds, updated at 10-millisecond intervals. A clock update is triggered when a register, loaded with a predefined value, has decremented to zero. Upon reaching zero, an interrupt is triggered that reloads the register, thus repeating the process.

The smaller the value loaded into this register, the more quickly it reaches zero and triggers an update. The clock runs more quickly in such an instance. A larger value means more time between updates; therefore, the clock runs more slowly. A **clock tick** is the amount of time between clock updates.

12.1.4 How NTP Makes Adjustments to System Time

Once NTP has selected a suitable synchronization source, NTP compares the source's time with that of the local clock. If NTP determines that the local clock is running ahead of or behind the synchronization source, NTP uses a general drift mechanism to slow down or speed up the clock as needed. NTP accomplishes this by issuing a series of new clock ticks. For example, if NTP detects that the local clock is drifting ahead by +0.1884338 second, it issues a series of new ticks in an effort to reduce the difference between the synchronization source and the local clock.

If the local system time is not reasonably correct, NTP will not set the local clock. For example, if the new time is more than 1000 seconds off in either direction, NTP does not set the clock. In this case, NTP logs the error and shuts down.

NTP maintains a record of the resets it makes along with informational messages in the NTP log file, TCPIP$NTP_RUN.LOG. See Section 12.6 for more details about event logging and help in interpreting an NTP log file.

12.1.5 Configuring the Local Host

As the system manager of the local host, you determine which network hosts to use for synchronization and populate an NTP configuration file with a list of the participating hosts.

NTP hosts may be configured in one or more of the following modes:

- Client/server mode

  This mode indicates that the local host wants to obtain time from the remote server *and is willing* to supply time to the remote server if necessary. This mode is appropriate in configurations involving a number of redundant time servers interconnected through diverse network paths. Internet time servers generally use this mode.

  Indicate this mode with a `peer` declaration in the configuration file. For example:

  ```
  peer 18.72.0.3
  ```

- Client mode

  This mode indicates that the local host wants to obtain time from the remote server *but it is not willing* to provide time to the remote server. Client mode is appropriate for file server and workstation clients that do not provide synchronization to other local clients. A host with higher stratum generally uses this mode.

Indicate client mode with the `server` declaration in the configuration file. For example:

```
server 18.72.0.3
```

- Broadcast mode

  This mode indicates that the local server will send periodic broadcast messages to a client population at the broadcast/multicast address specified. This specification normally applies to the local server operating as a sender.

  Indicate this mode with a broadcast declaration in the configuration file. For example:

```
broadcast 18.72.0.255
```

12.2 NTP Service Startup and Shutdown

The NTP service can be shut down and started independently of TCP/IP Services. The following files are provided:

- SYS$STARTUP:TCPIP$NTP_STARTUP.COM allows you to start up the NTP service independently.
- SYS$STARTUP:TCPIP$NTP_SHUTDOWN.COM allows you to shut down the NTP service independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$NTP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NTP service is started.
- SYS$STARTUP:TCPIP$NTP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NTP service is shut down.

12.3 Configuring Your NTP Host

The NTP configuration file TCPIP$NTP.CONF contains a list of hosts your system will use for time synchronization. Before configuring your host, you must:

1. Select time sources.
2. Obtain the IP addresses or host names of the time sources.
3. Obtain the version number of NTP that the hosts are running.

To ensure reliable synchronization, select multiple time sources that you are certain provide accurate time and are synchronized to an Internet time server.

To minimize common points of failure, avoid synchronizing:

- The local host to another peer at the same stratum unless the latter is receiving time from a lower stratum source to which the local host cannot connect
- More than one host in a particular administrative domain to the same time server outside that domain

To simplify configuration file maintenance, avoid configuring peer associations with higher stratum servers.

12.3.1 Creating the Configuration File

To create a configuration file for your local host, edit a copy of the file TCPIP$NTP.TEMPLATE (located in SYS$SPECIFIC:[TCPIP$NTP]) to add the names of participating hosts, then save the file as SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP.CONF. This file is not overwritten when you install subsequent versions of TCP/IP Services.

Note

If you had a previous version of NTP configured on your system, your TCPIP$NTP.CONF file is created automatically and is populated with entries from the file UCX$NTP.CONF when you run the TCPIP$CONFIG procedure.

12.3.2 Configuration Statements and Options

NTP configuration statements are formatted as follows:

- `peer` *`address`* `[key` *`ID`*`] [version` *`number`*`] [prefer] [minpoll` *`interval`*`] [maxpoll` *`interval`*`]`

  `server` *`address`* `[key` *`ID`*`] [version` *`number`*`] [prefer ]`

  `broadcast` *`address`* `[key` *`ID`*`] [version` *`number`*`] [ttl` *`nn`*`]`

  The following table describes the options to these statements:

Option	Description
`key` *`ID`*	For all packets sent to the address, includes authentication fields encrypted using the specified key identifier, an unsigned 32-bit integer. The default is no encryption.
`version` *`number`*	Specifies the version number to be used for outgoing NTP packets. Versions 1, 2, and 3 are the choices. The default is 3.
`prefer`	Marks the server as preferred. This host will be chosen for synchronization among a set of correctly operating hosts.
`minpoll` *`interval`*	Specifies the minimum polling interval for NTP messages, in seconds to the power of two. The allowable range is 4 (16 seconds) to 14 (16384 seconds), inclusive. This option is not applicable to reference clocks. The default is 6 (64 seconds).
`maxpoll` *`interval`*	Specifies the maximum polling interval (in seconds), for NTP messages. The allowable range is 4 (16 seconds) to 14 (16384 seconds) inclusive. The default is 10 (1024 seconds). (This option does not apply to reference clocks.)
`ttl` *`nn`*	Specifies the time-to-live for multicast packets. Used only with broadcast mode.

- `broadcastclient` *`address`*

  This statement directs the local server to listen for broadcast messages at the broadcast address of the local network. The default address is the subnet address with each host file bit set to 1. Upon hearing a broadcast message for the first time, the local server measures the nominal network delay using a brief client/server exchange with the remote server, then enters the `broadcastclient` mode, in which it listens for and synchronizes to succeeding broadcast messages. Note that, to avoid accidental or malicious disruption in

this mode, both the local and remote servers should use authentication and the same trusted key and key identifier.

- `multicastclient` *address*

  This statement directs the local server to listen for multicast messages at the group address of the global network. This command operates like the `broadcastclient` command but uses IP multicasting.

- `driftfile` *file-specification*

  This statement specifies the name of the file used to record the frequency offset of the local clock oscillator. If the file exists, it is read at startup to set the initial frequency offset, and then is updated hourly with the current frequency computed by the NTP server.

  If the file does not exist or if the `driftfile` command is not specified in the configuration file, the initial frequency offset is assumed to be zero. If the file does not exist but the `driftfile` keyword is specified without a parameter, the default, SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP.DRIFT, is used.

  In these cases, it may take some hours for the frequency to stabilize and the residual timing errors to subside.

  The drift file TCPIP$NTP.DRIFT consists of a single floating-point number, which records the frequency of the offset measured in parts per million (ppm).

- `enable auth | bclient | monitor | pll | stats`

  `disable auth | bclient | monitor | pll | stats`

  These statements enable and disable the following server options:

`auth`	Controls synchronization with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier. By default, `auth` is enabled.
`bclient`	Controls the server to listen for messages from broadcast or multicast servers. By default, `bclient` is disabled.
`monitor`	Controls the monitoring facility. By default, `monitor` is enabled.
`pll`	Controls whether the server adjusts its local clock by means of NTP. If disabled, the local clock free-runs at its intrinsic time and frequency offset. This flag is useful if the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients. In this case, the local clock driver is used. By default, `pll` is enabled.
`stats`	Enables the statistics facility. By default, `stats` is enabled.

12.3.2.1 NTP Monitoring Options

TCP/IP Services NTP includes a comprehensive monitoring facility suitable for continuous, long term recording of server and client timekeeping performance. See the `statistics` command below for a listing and example of each type of statistics currently supported. Statistic files are managed using file generation sets and scripts.

You can specify the following monitoring commands in your configuration file:

- `statistics` *name* `[ ... ]`

  Enables writing of statistics records. The following is a list of the supported *name* statistics:

  - `loopstats`

Enables recording of loop filter statistics information. Each update of the local clock outputs a line of the following form to the file generation set named loopstats:

```
48773 10847.650 0.0001307 17.3478 2
```

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). (A Julian Day [JD] begins at noon and runs until the next noon. The JD number is the number of days [or part of a day] since noon [UTC] on January 1, 4713 B.C. A Modified Julian Day [MJD] is the JD minus 2,400,000.5.)

The next three fields show time offset (in seconds), frequency offset (in parts per million) and time constant of the clock-discipline algorithm at each update of the clock.

- peerstats

  Enables recording of peer statistics information. This includes statistics records of all peers of an NTP server and of special signals, where present and configured. Each valid update appends a line of the following form to the current element of a file generation set named peerstats:

  ```
  48773 10847.650 127.127.4.1 9714 -0.001605 0.00000 0.00142
  ```

  The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next two fields show the peer address in dotted-quad notation and status, respectively. The status field is encoded in hexadecimal in the format described in Appendix A of the NTP specification (RFC 1305). The final three fields show the offset, delay, and dispersion (in seconds).

- clockstats

  Enables recording of clock driver statistics information. Each update received from a clock driver outputs a line in the following form to the file generation set named clockstats:

  ```
  49213 525.624 127.127.4.1 93 226 00:08:29.606 D
  ```

  The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next field shows the clock address in dotted-quad notation, The final field shows the last timecode received from the clock in decoded ASCII format, where meaningful. In some clock drivers, a good deal of additional information can be gathered and displayed as well. See information specific to each clock for further details.

- rawstats

  Enables recording of raw timestamps. Each valid update appends a line in the following form to the file generation set named rawstats:

  ```
  51554 79509.68  16.20.208.53  16.20.208.97
  3156617109.664603  3156617109.673268  3156617109.673268 31
  56617109.673268  3156617109.666556
  ```

  The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next two fields show the peer and local addresses in dotted-quad notation. The next four fields are:

  * The originate timestamp
  * The received timestamp

  - The transmitted timestamp (the last one sent to the same peer)
  - The timestamp of the packet's arrival on the server

- statsdir *directory-path*

  Indicates the full path of a directory where statistics files should be created.

12.3.2.2 Sample NTP Configuration File

A sample of the NTP configuration template follows:

```
#             Copyright 2000 Compaq Computer Corporation
#
#                  Example NTP Configuration File
#
# Rename this template to TCPIP$NTP.CONF.
#
# See the Compaq TCP/IP Services for OpenVMS Management manual for
# additional commands and detailed instructions on using this
# configuration file.
#
# The Network Time Protocol (NTP) provides synchronized timekeeping among
# a set of distributed time servers and clients. The local OpenVMS host
# maintains an NTP configuration file, TCPIP$NTP.CONF, of participating peers.
# TCPIP$NTP.CONF is maintained in the SYS$SPECIFIC:[TCPIP$NTP] directory.
#
# As the system manager populating this file, you must determine the
# peer hosts with which the local hosts should negotiate and synchronize.
# Include at least one (but preferably three) hosts that you are
# certain have the following characteristics:
#
#       * provide accurate time
#       * synchronize to Internet Time Servers (if they are not themselves
#         Internet Time Servers)
#
# The NTP configuration file is not dynamic, and therefore requires
# restarting NTP  after being edited to make the changes take effect.
# However, you can make run-time configuration requests interactively
# using the TCPIP$NTPDC utility.

# Your NTP configuration file should always include the following
# driftfile entry.  The driftfile is the name of the file that stores
# the clock drift (also known as frequency error) of the system clock.

driftfile SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP.DRIFT

# Sample peer entries follow.  Replace them with your own list of hosts
# and identify the appropriate association mode.  If you specify
# multiple hosts, NTP can choose the best source with which to
# synchronize. This also provides reliability in case one of the hosts
# becomes unavailable.

# Identify each peer with a fully qualified DNS host name or with
# an IP address in dotted-quad notation.

peer 18.72.0.3
peer 130.43.2.2
peer 16.1.0.22
peer parrot

# The following commands allow interoperation of NTP with another time service
# such as DTSS.  If enabled (by removing #), NTP will not set the system clock.

# server 127.127.1.0 prefer
# fudge 127.127.1.0 stratum 0
```

```
# The following commands allow this node to act as a backup NTP server (or as
# the sole NTP server on an isolated network), using its own system clock as
# the reference source.  If enabled (by removing #), this NTP server will
# become active only when all other normal synchronization sources are
# unavailable.

# server 127.127.1.0
# fudge 127.127.1.0 stratum 8
```

12.3.3 Using NTP with Another Time Service

A local host may run more than one time service. For example, a host may have both NTP and DTSS (Digital Time Synchronization Service) installed. However, only one of these time services is allowed to set the system clock.

If you are running a time service in addition to NTP, you must stop either the other time source or NTP from setting the system clock. You can stop NTP from setting the system clock by adding the following statements to the configuration file:

```
server 127.127.1.0 prefer
fudge 127.127.1.0 stratum 0
```

In these statements, the hardware address of the local clock (LOCAL) is 127.127.1.0. These statements force NTP to use its own system clock as a reference clock. The host continues to respond to NTP time queries but does not make any adjustments to the system clock, thereby allowing the other time service to make those changes.

12.4 Configuring NTP as Backup Time Server

You can configure the NTP service as a backup time server. In this case, if all other network synchronization sources become unavailable, the NTP service becomes active. You can also use this method to allow the local node to act as the NTP server in an an isolated network. To configure the NTP service as the backup server or the sole NTP server, enter the following commands in the NTP configuration file:

```
server 127.127.1.0
fudge 127.127.1.0 stratum 8
```

In this example, the stratum is set to a high number (8) so that it will not interfere with any other, possibly better, time synchronization source. You should set the stratum to a number that is higher than the stratum of all other time synchronization sources.

12.5 Operating with Time Zone Offsets

The operating system's installation procedure provides a command procedure that defines a time zone differential (offset) logical name in the system logical name table (LNM$SYSTEM_TABLE). The procedure is SYS$EXAMPLES:DAYLIGHT_SAVINGS.COM. The logical name is SYS$TIMEZONE_DIFFERENTIAL.

To change the time zone differential offset, follow these steps:

1. Make sure the following logical name is defined:

   ```
   SYS$TIMEZONE_DIFFERENTIAL
   ```

   This logical name is defined automatically when you install the OpenVMS operating system.

2. Run the command procedure SYS$COMMON:[SYSMGR]UTC$CONFIGURE_TDF.
3. Select an option to set the time differential factor.

   The procedure prompts you for the time differential factor (TDF) (the difference between your system time and Universal Coordinated Time (UTC)). Specify the difference in *hh:mm* format.

   North and South America have negative offsets from UTC. Europe, Africa, Asia, and Australia all have positive offsets. Enter the time differential factor.
4. The procedure asks whether or not you want to modify the local system time.

   Answer Yes or No.

   The procedure defines the system logical name SYS$TIMEZONE_DIFFERENTIAL to be the system time differential factor (or time zone offset). For example, during the summer months in Boston, the procedure defines SYS$TIMEZONE_DIFFERENTIAL as -14400 seconds.
5. If NTP is enabled, follow these additional steps:

   Stop NTP by entering the following command:

   ```
   $ @SYS$STARTUP:TCPIP$NTP_SHUTDOWN.COM
   ```

   Restart NTP by entering the following command:

   ```
   $ @SYS$STARTUP:TCPIP$NTP_STARTUP.COM
   ```

   Note

   NTP works with UTC only. However, the OpenVMS time reflects the local time. Therefore, you must follow the preceding steps to account for a change in daylight saving time (DST).

12.6 NTP Event Logging

NTP maintains a record of system clock updates in the file SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP_RUN.LOG. NTP reopens this log file daily, each time creating a new version of the file (older versions are not automatically purged). Events logged to this file may include the following messages:

- Synchronization status that indicates synchronization occurred, was lost, was reestablished, stratum changes, and so on.
- System time adjustments
- Time adjustment status
- Packet transmission status

To set the amount of logging information to be recorded, set the following logical name to a value from 1 through 6, where 6 specifies the most detailed logging:

```
TCPIP$NTP_LOG_LEVEL n
```

Table 12–1 describes the messages most frequently included in the NTP log file.

Table 12–1 NTP Log File Messages

Message	Description
Synchronized to *IP-address*	Announces that a peer candidate has passed validity and accuracy tests (as performed by the clock selection algorithms) and has been selected as the new synchronization source. For example: `synchronized to 16.20.208.100, stratum=2`
Time reset *time*	Indicates that NTP has set the local clock by slewing the local time to match the synchronization source. This happens because the local host is no longer synchronized. For example: `time reset (slew) -0.218843 sec`
Synchronization lost	This usually occurs after a time reset. All peer filter registers are cleared, for example, for that particular peer; all state variables are reset along with the polling interval; and the clock selection procedure is once again performed.
Previous time adjustment incomplete	Indicates that the last clock adjustment did not finish in one attempt. The residual is added to the next adjustment.
Couldn't resolve *hostname*, giving up on it	Indicates that the host name could not be resolved. This peer will not be considered for the candidate list of peers. For example: `couldn't resolve 'fred',  giving up on it`
Send to *IP-address*: *reason*	Indicates that a problem occurred while sending a packet to its destination. The most common *reason* logged is "connection refused." For example: `sendto(16.20.208.100): connection refused`
Connection reestablished to *IP-address*	Indicates that errors occurred when sending packets, but now packets are being sent successfully. For example: `connection reestablished to 16.20.208.100`
Time error *delta-time* is way too large (set clock manually)	NTP has detected a time difference greater than 1000 seconds between the local clock and the server clock. You must set the clock manually or use the NTPDATE program and then restart NTP. Once NTP sets the clock, it will continuously track the discrepancy between the local time and NTP time and adjust the clock accordingly.
offset : *n* sec freq: *x* poll: *y* sec	An hourly message, where: • `offset` is the offset (in seconds) of the peer clock relative to the local clock (that is, the amount to adjust the local clock to bring it into correspondence with the reference clock).

(continued on next page)

Table 12–1 (Cont.) NTP Log File Messages

Message	Description
	• `freq` is the computed error in the intrinsic frequency of the local clock (also known as "drift") (in parts per million). • `poll` indicates the minimum interval (in seconds) between transmitted messages (that is, messages sent between NTP peers, as in a client to a server).
No clock adjustments will be made, DTSS is active	Indicates that the DTSS time service is running on the system. The DTSS time service should be disabled if you would like NTP to set the system time. To disable the DTSS time service, enter the following command: `$ RUN SYS$SYSTEM:NCL DISABLE DTSS` Alternatively, you can configure the NTP server not to make clock adjustments, as described in Section 12.3.3. NTP dynamically detects whether the DTSS time service is enabled at any time and will log this message if appropriate.
Clock adjustments will resume. DTSS no longer active	Indicates that the DTSS time service has been disabled on the system. NTP will now handle clock adjustments. NTP dynamically detects whether the DTSS time service is enabled at any time and will log this message if appropriate.

12.6.1 Sample NTP Log File

The following sample shows an NTP log file:

```
16 Apr 16:36:30   ntpd version = 3-5.91
16 Apr 16:36:31   tickadj = 97, tick = 976, tvu_maxslew = 99231, est. hz = 1024
16 Apr 16:36:31   precision = 976 usec
16 Apr 16:36:33   read drift of 0 from TCPIP$NTP.DRIFT
16 Apr 16:43:00   synchronized to 16.20.208.100, stratum=2
16 Apr 16:43:00   time reset (slew) -62.810275 sec
16 Apr 16:43:00   synchronization lost
16 Apr 16:44:58   Previous time adjustment incomplete; residual -0.005758 sec
16 Apr 16:48:21   synchronized to 16.20.208.100, stratum=2
16 Apr 16:52:28   Previous time adjustment incomplete; residual -0.005270 sec
16 Apr 16:53:26   Previous time adjustment incomplete; residual -0.085888 sec
16 Apr 17:11:40   synchronized to 16.20.208.23, stratum=3
16 Apr 17:13:49   synchronized to 16.20.208.100, stratum=2
16 Apr 17:14:53   time reset (slew) -0.577109 sec
16 Apr 17:14:53   synchronization lost
16 Apr 17:21:38   synchronized to 16.20.208.23, stratum=3
16 Apr 17:26:54   synchronized to 16.20.208.100, stratum=2
16 Apr 17:35:24   offset: 0.033115 sec  freq: -7.813 ppm  poll: 1024 sec
16 Apr 17:46:23   synchronized to 16.20.208.97, stratum=3
16 Apr 17:47:28   Previous time adjustment incomplete; residual -0.000020 sec
16 Apr 17:49:32   Previous time adjustment incomplete; residual 0.093696 sec
16 Apr 17:49:36   Previous time adjustment incomplete; residual 0.003318 sec
16 Apr 17:52:08   Previous time adjustment incomplete; residual -0.049460 sec
16 Apr 17:52:24   Previous time adjustment incomplete; residual 0.003416 sec
16 Apr 17:53:28   Previous time adjustment incomplete; residual 0.000088 sec
16 Apr 18:06:10   time reset (slew) -0.218843 sec
16 Apr 18:06:11   synchronization lost
16 Apr 18:17:39   synchronized to 16.20.208.97, stratum=3
16 Apr 18:17:43   synchronized to 16.20.208.100, stratum=2
16 Apr 18:21:47   synchronized to 16.20.208.97, stratum=3
16 Apr 18:23:41   synchronized to 16.20.208.100, stratum=2
16 Apr 18:35:40   offset: -0.052522 sec  freq: -7.839 ppm  poll: 1024 sec
```

12.7 NTP Authentication Support

Authentication support is implemented using the MD5 algorithm to compute a message digest. The servers involved in an association must agree on the key and key identifier used to authenticate their messages.

Keys and related information are specified in a key file. Keys are used for:

- Ordinary NTP associations
- The NTPQ utility program
- The NTPDC utility program

12.7.1 NTP Authentication Commands

Table 12–2 describes additional configuration statements and options used to support authentication.

Table 12–2 Authentication Commands

Command	Description
keys *keys-file*	Specifies the file name for the keys file, which contains the encryption keys and key identifiers used by NTP, NTPQ, and NTPDC when operating in authenticated mode.
trustedkey *key-ID* [...]	Specifies the encryption key identifiers that are trusted for the purposes of authenticating peers suitable for synchronization, as well as keys used by the NTPQ and NTPDC programs. The authentication procedures require that the local and remote servers share the same *key-ID* and key value for this purpose, although different key values can be used with different servers. The *key-ID* arguments are 32-bit unsigned decimal integers from 1 to 15. Note that the NTP key 0 is used to indicate an invalid key value or key identifier; therefore, it should not be used for any other purpose.
requestkey *key-ID*	Specifies the key identifier to use with the NTPDC program, which uses a proprietary protocol specific to this implementation of NTP. This program is useful to diagnose and repair problems that affect the operation of NTP. For information about NTPDC, see Section 12.8.3. The *key-ID* argument to this command is an unsigned 32-bit decimal number that identifies the trusted key in the keys file. If no requestkey command is included in the configuration file, or if the keys do not match, any request to change a server variable is denied.
controlkey *key-ID*	Specifies the key identifier to use with the NTPQ program, which uses the standard protocol defined in RFC-1305. This program is useful to diagnose and repair problems that affect the operation of NTP. For more information about NTPQ, see Section 12.8.4. The *key-ID* argument to this command is a 32-bit decimal integer that identifies a trusted key in the keys file. If no controlkey command is included in the configuration file, or if the keys do not match, any request to change a server variable is denied.

Keys are defined in a keys file, as described in Section 12.7.2.

12.7.2 Authentication Key Format

The NTP service reads keys from a keys file that is specified using the keys command in the configuration file. You can supply one or more keys from 1 to 15 in the keys file.

Key entries use the following format:

```
key-ID key-type key-value
```

The fields are:

- *key-ID*, which is an arbitrary, unsigned 32-bit number (in decimal). The range of possible values is 1 to 15. Key IDs are specified by the `requestkey` and `controlkey` statements in the configuration file. The key ID number 0 (56 zero bits) is reserved; it is used to indicate an invalid key ID or key value.
- *key-type*, which identifies the type of key value. Only one key format, "M," is currently supported. This indicates that the MD5 authentication scheme is being used.
- *key-value*, which is an ASCII string of one to eight characters. The following characters are not allowed:

  space
  pound sign (#)
  \ t
  \ n
  \ 0

Because this file contains authorization data, Compaq recommends that you limit read access to this file. In particular, you should disable world read access.

The following is a sample keys file:

```
#
#
4       M    DonTTelL
6       M    hElloWrl
12      M    ImASecrt
```

12.8 NTP Utilities

NTP provides several utility programs that help you manage and make changes to the NTP server. These utilities include:

- TCPIP$NTPDATE, the date and time utility that sets the local date and time by polling the specified server. Run NTPDATE manually or from the host startup script to set the clock at boot time before NTP starts.

  NTPDATE will not set the date if NTP is already running on the same host.

  For information about using NTPDATE, see Section 12.8.1.
- TCPIP$NTPTRACE, the trace utility that follows the chain of NTP servers back to their master time source. For information about using NTPTRACE, see Section 12.8.2.
- TCPIP$NTPDC, the special query program that provides extensive state and statistics information and allows you to set configuration options at run time. Run this program in interactive mode or with command line arguments.

  For information about using NTPDC, see Section 12.8.3.
- TCPIP$NTPQ, the standard query program that queries NTP servers about their current state and requests changes to that state. For information about using NTPQ, see Section 12.8.4.

12.8.1 Setting the Date and Time with NTPDATE

The NTPDATE program sets the local date and time by polling a specified server or servers to determine the correct time. A number of samples are obtained from each of the servers specified, and a subset of the NTP clock filter and selection algorithms are applied to select the best samples. The accuracy and reliability of NTPDATE depends on the number of servers it polls, the number of polls it makes each time it runs, and the interval length between runs.

Run NTPDATE manually to set the host clock or from the host startup file to set the clock at boot time. It is useful in some cases to set the clock manually before you start NTP. NTPDATE makes time adjustments (called "stepping the time") by calling the OpenVMS routine SYS$SETIME.

Note

NTPDATE will not set the date if an NTP server is running on the same host.

Table 12–3 describes the NTPDATE command options. To use these options, define the NTPDATE command, as follows:

```
NTPDATE:==$SYS$SYSTEM:TCPIP$NTPDATE.EXE
```

Enter commands using the following format:

```
NTPDATE [option...] host [host...]
```

For example, the following command sets the clock based on the time provided from one of the specified hosts (BIRDY, OWL, or FRED):

```
$ NTPDATE BIRDY OWL FRED
```

NTP sets the date and time by polling the servers you specify as arguments to the command. Samples are obtained from each of the specified servers. NTP then analyzes the results to select the best server to use as a time source.

Table 12–3 NTPDATE Options

Option	Description
`-d`	Changes the time and prints information useful for debugging.
`-o` *version*	Specifies the NTP version (1 or 2) for outgoing packets (for compatibility with older versions of NTP). Version 3 is the default.
`-p` *n*	Specifies the number of samples NTPDATE acquires from each server. The default is four. You can specify from one to eight.
`-q`	Specifies a query only; does not set the clock.

12.8.2 Tracing a Time Source with NTPTRACE

Use the NTPTRACE utility to determine the source from which an NTP server obtains its time. NTPTRACE follows the chain of time servers back to the master time source.

To run NTPTRACE, define a foreign command as follows:

```
$ NTPTRACE:==$SYS$SYSTEM:TCPIP$NTPTRACE.EXE
```

Use the following syntax when entering commands:

```
NTPTRACE [option...]
```

The following example shows output from an NTPTRACE. In this example, the chain of servers from the local host to the stratum 1 server FRED, which is synchronizing to a GPS reference clock.

```
$ NTPTRACE

LOCALHOST: stratum 3, offset -0.000000, synch distance1.50948
parrot.birds.com: stratum 2, offset -0.126774, synch distance 0.00909
fred.birds.com: stratum 1, offset -0.129567, synch distance 0.00168,
refid 'GPS'
```

All times are in seconds. The output fields on each line are as follows:

- Host name
- Host's stratum
- Time offset between the host and the local host (not always zero for LOCALHOST).
- Synchronization distance
- Reference clock ID (only for stratum-1 servers)

Table 12–4 describes the NTPTRACE command options.

Table 12–4 NTPTRACE Options

Option	Description
`-d`	Enables debugging output.
`-n`	Displays IP addresses instead of host names. This may be necessary if a name server is down.
`-r` *retries*	Sets the number of retransmission attempts for each host. The default is 5.
`-t` *timeout*	Sets the retransmission timeout (in seconds). The default is 2.
`-v`	Displays additional information about the NTP servers.

12.8.3 Making Run-Time Requests with NTPDC

Section 12.3 discussed how to use the configuration file to configure NTP on your system. In addition to using a configuration file, you can make run-time changes to NTP with query commands by running the NTPDC utility. NTPDC displays time values in seconds.

Run-time requests are always authenticated requests. Authentication provides verification that the requester has permission to make such changes but also gives an extra degree of protection against transmission errors.

The reconfiguration facility works well with a server on the local host and between time-synchronized hosts on the same LAN. The facility works poorly for more distant hosts. Authenticated requests include a timestamp. The server compares the timestamp to its *receive* timestamp. If they differ by more than a small amount, the request is rejected. This is done for two reasons:

- It makes it more difficult for an intruder to overhear traffic on your LAN.

- It makes it more difficult for topologically remote hosts to request configuration changes to your server.

To run NTPDC, enter the following command:

```
$ RUN SYS$SYSTEM:TCPIP$NTPDC.EXE
```

At the NTPDC> prompt, enter the appropriate type of command from the following list:

- Interactive commands
- Control commands
- Run-time configuration request commands

12.8.3.1 NTPDC Interactive Commands

Interactive commands consist of a command name followed by one or more keywords. The interactive commands are:

- `help [ command_keyword ]`

  Enter a question mark (?) to display a list of all the command keywords known to this version of NTPDC. Enter a question mark followed by a command keyword to display information about the function and use of the command.

- `host hostname`

  Set the host to which future queries will be sent. The *hostname* can be either a host name or a numeric address.

- `hostnames [ yes | no ]`

  If you specify `yes`, host names are displayed. If you specify `no`, numeric addresses are displayed instead. The default is `yes` unless you include the `-n` option on the command line, as described in Table 12–4.

- `keyid key-ID`

  This command allows the specification of a key number to be used to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose.

- `quit`

  Exits NTPDC.

- `passwd`

  This command prompts you to type in a password (not echoed) that will be used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose if such requests are to be successful.

- `timeout milliseconds`

  Specify a timeout period for responses to server queries. The default is about 8000 milliseconds (8 seconds). Because NTPDC retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

12.8.3.2 NTPDC Control Message Commands

Control message commands request information about the server. These are read-only commands in that they make no modification of the server configuration state.

The NTPDC control message commands are:

- `listpeers`

  Displays a brief list of the peers for which the server is maintaining state. These include all configured peer associations as well as those peers whose stratum is such that they are considered by the server to be possible future synchonization candidates.

- `peers`

  Obtains a list of peers for which the server is maintaining state, along with a summary of that state. The summary information includes:

  - The address of the remote peer
  - The local interface address (0.0.0.0 if a local address has not been determined)
  - The stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized)
  - The polling interval (in seconds)
  - The reachability register (in octal)
  - The current estimated delay, offset, and dispersion of the peer (in seconds)

  In addition, the character in the left margin indicates the mode this peer entry is operating in, as follows:

  Plus sign (+) denotes symmetric active.
  Minus sign (-) indicates symmetric passive.
  Equals sign (=) means the remote server is being polled in client mode.
  Up arrow (^) indicates that the server is broadcasting to this address.
  Tilde (~) denotes that the remote peer is sending broadcasts.
  Asterisk (*) marks the peer to which the server is currently synchronizing.

  The contents of the host field may be one of four forms. It may be:

  - Host name
  - IP address
  - Reference clock implementation name with its parameter
  - `REFCLK` (*`implementation number parameter`*).

  If you specify `hostnames no`, only IP addresses will be displayed.

- `dmpeers`

  Displays a slightly different peer summary list, identical to the output of the `peers` command, except for the character in the leftmost column. Characters only appear beside peers that were included in the final stage of the clock selection algorithm.

  Dot (.) indicates that this peer was rejected in the falseticker detection.
  Plus sign (+) indicates that the peer was accepted.

Asterisk (*) denotes the peer to which the server is currently synchronizing.

- showpeer *peer_address* [...]

  Shows a detailed display of the current peer variables for one or more peers.

- pstats *peer_address* [...]

  Shows per-peer statistics counters associated with the specified peers.

- loopinfo [oneline multiline]

  Displays the values of selected loop-filter variables. The loop filter is the part of NTP that adjusts the local system clock. These options include:

  – offset — the last offset given to the loop filter by the packet processing code.

  – frequency — the frequency error of the local clock (in parts per million)

  – time_const — controls the stiffness of the phase-lock loop and thus the speed at which it can adapt to oscillator drift.

  – watchdog timer value — the number of seconds that have elapsed since the last sample offset was given to the loop filter.

  The oneline and multiline options specify the format in which this information is to be displayed; multiline is the default.

- sysinfo

  Displays a variety of system state variables, such as the state related to the local server.

  The system flags show various system flags, some of which can be set and cleared by the enable and disable configuration commands, respectively. These are the auth, bclient, monitor, pll, pps and stats flags.

  The stability is the residual frequency error remaining after the system frequency correction is applied. It is intended for maintenance and debugging.

  The broadcastdelay shows the default broadcast delay, as set by the broadcastdelay configuration command.

  The authdelay shows the default authentication delay, as set by the authdelay configuration command.

- sysstats

  Displays statistics counters maintained in the protocol module.

- memstats

  Displays statistics counters related to memory allocation code.

- iostats

  Displays statistics counters maintained in the input/output module.

- timerstats

  Displays statistics counters maintained in the timer/event queue support code.

- reslist

  Displays the server's restriction list. This list is displayed in the order in which the restrictions are applied.

- monlist [*version*]

  Displays traffic counts collected. This is maintained by the monitor facility. Normally, you should not need to specify the version number.

12.8.3.3 NTPDC Request Commands

The following commands make authenticated requests:

- addpeer *peer-address key-ID* [*version*] [*prefer*]

  Adds a configured peer association at the given address and operates in symmetric active mode. The existing association with the same peer may be deleted when this command is executed or may be converted to conform to the new configuration.

  The *key-ID* is the key identifier for requestkey, as described in Table 12–2. All outgoing packets to the remote server will have an authentication field attached that is encrypted with this key.

  The value for *version* can be 1, 2, or 3. The default is Version 3.

  The *prefer* keyword indicates a preferred peer that will be used for clock synchronization, if possible.

- addserver *peer-address key-ID* [*version*] [*prefer*]

  This command is the same as addpeer except that the operating mode is client.

- broadcast *peer-address key-ID* [*version*] [*prefer*]

  This command is the same as addpeer, except that the operating mode is broadcast. In this case, a valid key identifier and key value are required. The *peer-address* parameter can be the broadcast address of the local network or a multicast group address assigned to NTP.

- unconfig *peer-address* [...]

  Causes the configured bit to be removed from the specified remote peer. This deletes the peer association. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue in this fashion.

- enable [*flag*] [...]

  disable [*flag*] [...]

  These commands operate in the same way as the enable and disable configuration commands. See Section 12.3.2.

- fudge *peer-address* [*time1*] [*time2*] [stratum *stratum*] [*refID*]

  Provides a way to set time, stratum, and identification data for a reference clock. (The TCP/IP Services product supports only the local reference clock.)

You can also run NTPDC by defining a foreign command as follows:

```
$ NTPDC:==$SYS$SYSTEM:TCPIP$NTPDC.EXE
```

Use the following syntax when entering commands:

```
NTPDC [option...]
```

Table 12–5 describes the NTPDC options.

Table 12–5 NTPDC Options

Option	Description
`-c` *command*	The command argument is interpreted as an interactive format command and is added to the list of commands to be executed on the specified hosts. Multiple `-c` options may be given.
`-i`	Forces NTPDC to operate in interactive mode.
`-l`	Obtains a list of peers that are known to the servers.
`-n`	Displays all host addresses in numeric format rather than converting them to host names.
`-p`	Displays a list of the peers known to the server as well as a summary of their state.
`-s`	Displays a list of the peers known to the server as well as a summary of their state, but in a slightly different format than the `-p` option.

12.8.4 Querying the NTP Server with NTPQ

The NTPQ program allows you to query the NTP server about its current state and request changes to that state. NTPQ can also obtain and display a list of peers in a common format by sending multiple queries to the server.

The NTPQ program authenticates requests based on the key entry in the keys file that is configured using the `controlkey` command, as described in Table 12–2.

The NTPQ program uses NTP mode 6 packets to communicate with the NTP server; therefore, it is used to query any compatible server on the network. Because NTP is a UDP protocol, this communication is somewhat unreliable over long distances (in terms of network topology). The NTPQ program makes one attempt to restransmit requests and times out requests if the remote host does not respond within the expected amount of time. NTPQ displays time values in milliseconds.

To run the NTPQ program, enter the following command:

```
$ RUN SYS$SYSTEM:TCPIP$NTPQ.EXE
```

At the NTPQ> prompt, enter commands using the following syntax:

command [*options...*]

The following commands allow you to query and set NTP server state information:

- ? [*command_keyword*]

  A question mark (?) by itself prints a list of all the command keywords known to this version of NTPQ. A question mark followed by a command keyword prints function and usage information about the command.

- `addvars` *variable_name*[`=`*value*] [`,...`]
- `rmvars` *variable_name* [`,...`]
- `clearvars`

  The data carried by NTP mode 6 messages consists of a list of items in the form:

  variable_name=value

In requests to the server to read variables, the *=value* portion is ignored and can be omitted. The NTPQ program maintains an internal list in which data to be included in control messages can be assembled and sent using the `readlist` and `writelist` commands. The `addvars` command allows variables and their optional values to be added to the list. If more than one variable is to be added, the list should be separated by commas and should not contain blank spaces. The `rmvars` command can be used to remove individual variables from the list, while the `clearlist` command removes all variables from the list.

- `authenticate yes | no`

  By default, NTPQ does not authenticate requests unless they are write requests. The `authenticate yes` command causes NTPQ to send authentication with all requests it makes. Authenticated requests cause some servers to handle requests slightly differently. To prevent any mishap, do a peer display before turning on authentication.

- `cooked`

  Reformats variables that are recognized by the server. Variables that NTPQ does not recognize are marked with a trailing question mark (?).

- `debug more | less | no`

  Adjusts level of NTPQ debugging. The default is `debug no`.

- `help`

  Displays the list of NTPQ interactive commands. This is the same as question mark (?).

- `host [`*host-name*`]`

  Sets the host to which future queries will be sent; *host-name* may be either a host name or an Internet address. If *host-name* is not specified, the current host is used.

- `hostnames yes | no`

  If `yes` is specified, displays host names in information displays. If `no` is specified, displays Internet addresses instead. The default is `hostnames yes`. The default can be modified using the command line option `-n`.

- `key-ID` *n*

  Specifies the key ID number to be used to authenticate configuration requests. This must correspond to a key ID number the server has been configured to use for this purpose (see Section 10.7.2).

- `keytype md5 | des`

  Sets the authentication key to either MD5 or DES. Only MD5 is supported in this implementation.

- `ntpversion 1 | 2 | 3`

  Sets the NTP version number that NTPQ claims in packets. Default is 3. Mode 6 control messages (and modes, for that matter) did not exist in NTP version 1.

- `passwd`

  Prompts you to enter a password (not echoed) that is used to authenticate configuration requests. The password must correspond to the key value configured for use by the NTP server for this purpose if such requests are to be successful (see Section 12.7.2).

- `quit`

  Exits NTPQ.

- `raw`

  Displays all output from query commands as received from the remote server. The only data formatting performed is to translate non-ASCII data into a printable form.

- `timeout` *`milliseconds`*

  Specifies a timeout period for responses to server queries. The default is about 5000 milliseconds. Since NTPQ retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value.

12.8.4.1 NTPQ Control Message Commands

Each peer known to an NTP server has a 16-bit integer association identifier assigned to it. NTP control messages that carry peer variables must identify the peer that the values correspond to by including the peer's association ID. An association ID of zero indicates the variables are system variables whose names are drawn from a separate name space.

Control message commands result in one or more NTP mode 6 messages being sent to the server, and cause the data returned to be displayed in a format that you control using the commands listed in Section 12.8.4. Most commands send a single message and expect a single response. The exceptions are the `peers` command, which sends a preprogrammed series of messages to obtain the data it needs, and the `mreadlist` and `mreadvar` commands, which are repeated for each specified association.

- `associations`

  Displays a list of association identifiers and peer status for recognized peers of the server being queried. The list is printed in columns. The first of these is an index numbering the associations from 1 for internal use; the second is the actual association identifier returned by the server; and the third is the status word for the peer. This is followed by a number of columns containing data decoded from the status word. The data returned by the associations command is cached internally in NTPQ. The index is then used when dealing with servers that use association identifiers. For any subsequent commands that require an association identifier as an argument, the form `index\\` may be used as an alternative.

- `lassociations`

  Obtains and displays a list of association identifiers and peer status for all associations for which the server is maintaining state. This command differs from the `associations` command only for servers which retain state for out-of-spec client associations. Such associations are normally omitted from the display when the `associations` command is used but are included in the output of the `lassociations` command..

- lopeers

  Obtains and displays a list of all peers and clients having the destination address.

- lpassociations

  Displays data for all associations, including unrecognized client associations, from the internally cached list of associations.

- lpeers

  Similar to peers except that a summary of all associations for which the server is maintaining state is displayed. This command can produce a much longer list of peers.

- mreadlist *assocID assocID*

  Similar to the readlist command except that the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent associations command.

- mreadvar *assocID assocID* [*variable_name*[=*value*] [,...]]

  Similar to the readvar command except that the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent associations command.

- opeers

  An old form of the peers command, with the reference ID replaced by the local interface address.

- passociations

  Displays association data concerning recognized peers from the internally cached list of associations. This command performs identically to the associations command except that it displays the internally stored data rather than make a new query.

- peers

  Displays a list of recognized peers of the server, along with a summary of each peer's state. Summary information includes the address of the remote peer; the reference ID (0.0.0.0 if the reference ID is unknown); the stratum of the remote peer; the polling interval (in seconds); the reachability register (in octal); and the current estimated delay, offset, and dispersion of the peer (in milliseconds).

  The character in the left margin indicates the fate of this peer in the clock selection process. The codes are as follows:

  Space indicates that the peer was discarded, because of high stratum or failed sanity checks.
  Lowercase x indicates that the peer was designated a falseticker by the intersection algorithm.
  Dot (.) indicates that this peer was culled from the end of the candidate list.
  Hyphen (-) indicates that the peer was discarded by the clustering algorithm.
  Plus sign (+) indicates that the peer was included in the final selection set.
  Pound sign (#) indicates that the peer was selected for synchronization, but the distance exceeds the maximum.
  Asterisk (*) indicates that the peer was selected for synchronization.

Since the peers command depends on the ability to parse the values in the responses it gets, it might fail to work with servers that poorly control the data formats.

The contents of the host field may in be one of four forms: a host name, an IP address, a reference clock implementation name with its parameter, or REFCLK (implementation number parameter). If you specified hostnames no, the IP addresses will be displayed.

- pstatus assocID

  Sends a read status request to the server for the given association. The names and values of the peer variables returned will be printed. The status word from the header is displayed preceding the variables, both in hexadecimal and in English.

- readlist [*assocID*]

  Requests that the server return the values of the variables in the internal variable list. If the association ID is omitted or is zero, the variables are assumed to be system variables. Otherwise, they are treated as peer variables. If the internal variable list is empty, a request is sent without data; the remote server should return a default display.

- readvar [*assocID*] [*variable_name*[=*value*] [,...]]

  Requests that the values of the specified variables be returned by the server by sending a read variables request. If the association ID is omitted or is given as zero, the variables are system variables; otherwise, they are peer variables, and the values returned are those of the corresponding peer. If the variable list is empty, a request is sent without data; the remote server should return a default display.

- showvars

  Displays the variables on the variable list.

- version

  Displays the NTPQ version number.

- writelist [assocID]

  Like the readlist request except that the internal list variables are written instead of read.

- writevar *assocID variable_name*=*value* [,...]

  Like the readvar request except that the specified variables are written instead of read.

You can also run NTPQ by defining a foreign command as follows:

```
NTPQ:==$SYS$SYSTEM:TCPIP$NTPQ.EXE
```

Use the following syntax when entering the NTPQ foreign command:

NTPQ [-i] [-n] [-p] [-c *command*] [*host1,host2,...*]

Table 12–6 describes the NTPQ options.

Table 12–6 NTPQ Options

Option	Description
`-c` *command*	Adds the specified interactive command to the list of commands to be executed on the specified host. You can enter multiple `-c` options on the command line.
`-i`	Forces NTPQ to operate in interactive mode. This is the default mode of operation.
`-n`	Displays host addresses numeric format rather than converting them to host names.
`-p`	Displays a list of the peers known to the server as well as a summary of their state.

The `-c` and `-p` options send the query to the specified host immediately. If you omit the host names, the default is the local host. To enter interactive mode, specify the `-i` or `-n` option.

12.9 Solving NTP Problems

Some common NTP problems include:

- Out of synch system clock.

  The NTP cannot synchronize a clock that is off by more than 1000 seconds. To solve this problem, set the clock using TCPIP$NTPDATE, then restart the NTP.

- NTPDATE fails to set the clock.

  This occurs if the NTP is already running.

- More than one service is actively setting the system clock.

  The NTP can run with other time services but must be explicitly instructed not to set the system clock. NTP can still provide synchronization to other clients even if it is not updating the system clock.

- NTP appears to be running without error, but the system clock is off by a one, two, three, or four-hour interval.

  You may need to adjust the time zone differential by running SYS$COMMON:[SYSMGR]UTC$CONFIGURE_TDF.COM. (See Section 12.5 for more information.)

13
Configuring SNMP

The Simple Network Management Protocol (SNMP) is network management technology that facilitates the management of a TCP/IP network or internet in a vendor-independent manner. SNMP enables a network administrator to manage the various network components using a set of well-known procedures understood by all components, regardless of the vendor that manufactured them.

Configuring SNMP on your OpenVMS system allows a remote SNMP management client to obtain information about your host and to set system and network parameters.

This chapter reviews key concepts of SNMP and describes:

- How to manage the SNMP service (Section 13.2)
- How to verify the installation of SNMP (Section 13.3)
- How to configure SNMP (Section 13.4)
- SNMP log files (Section 13.5)
- How to solve SNMP problems (Section 13.6)

For information about writing programs using SNMP, refer to the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

13.1 Key Concepts

Systems using SNMP are divided into two categories:

- Management consoles, sometimes called clients, network management stations, or directors
- Agents, sometimes called servers

The management console is the system that issues a query; the agents run on the system being queried. Queries are sent and received in the form of protocol data units (PDUs) inside SNMP messages, which are carried in user data protocol (UDP) datagrams.

You can configure your host so that an SNMP client can obtain information about your host and perform updates on your host's management information base (MIB) data items. For example, you can configure your host to:

- Respond to a client's read requests ("Gets") for network information.
- Process client write requests ("Sets") on your host's MIB data items.
- Send alert messages ("traps") to a client as a result of events that might need to be monitored (for example, an authentication failure).

TCP/IP Services provides an SNMP master agent, two subagents (MIB II and Host Resources MIB), a MIB converter and compiler, a simple MIB browser, and MIB utility programs. Each subagent contains routines that perform read and write operations on its MIB data items.

Table 13–1 describes the SNMP components and the sample code supplied for custom subagent development.

Table 13–1 SNMP Components

Component	Description
Master agent SNMP Version 2	Process name: TCPIP$SNMP_*n*, where *n* is the number of times that the master agent has been started since the SNMP service was enabled.
	Keeps track of managed objects and allows objects to register themselves. Sends information about these objects to remote SNMP management consoles. Also maintains a small set of variables for the MIB II component.
MIB II	Process name: TCPIP$OS_MIBS.
	Provides information about the TCP/IP protocol stack and other network activity.
Host resources MIB	Process name: TCPIP$HR_MIB.
	Provides information about the host system.
MIB converter	Extracts a MIB definition in ASN.1 notation into a MIB definition (.MY) file.
MIB compiler	Compiles MIB-definition files (for example, CHESS_MIB.MY) into source code templates for use in building subagents.
SNMP utility programs	Acts as a simple clients to obtain a set of values for a MIB and to listen for and send trap messages. For information about using the MIB utility programs, see the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.
SNMP subagent example	Implements an example based on the chess game; includes executable and source code.

13.1.1 Understanding How SNMP Operates

The TCPIP$CONFIG procedure sets up the SNMP UDP-based service at well-known port 161.

In addition, TCPIP$CONFIG sets up required files in the SYS$SYSDEVICE:[TCPIP$SNMP] directory.

The SNMP startup procedure (SYS$STARTUP:TCPIP$SNMP_STARTUP.COM) runs from the general TCPIP$STARTUP.COM procedure or can be run directly by the system manager.

TCPIP$SNMP_STARTUP.COM does the following:

1. Checks the TCP/IP Services license and enables the SNMP service.
2. Installs images with the required privileges (as appropriate: BYPASS, PHY_IO, and WORLD).
3. Runs SYS$STARTUP:TCPIP$SNMP_SYSTARTUP.COM.

To ensure compatibility with previous versions of TCP/IP Services, TCPIP$SNMP_SYSTARTUP.COM in turn runs SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$EXTENSION_MIB_STARTUP.COM, which installs and adjusts privileges for any additional, user-written subagents.

On startup, the TCP/IP Services kernel runs the TCPIP$SYSTEM:TCPIP$SNMP_RUN.COM procedure, which does the following:

- Purges log files in the SYS$SYSDEVICE:[TCPIP$SNMP] directory.
- Runs the subagent image as a detached process.
- Runs SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$EXTENSION_MIB_RUN.COM to start any additional subagents.

As each subagent starts, it makes itself known to the master agent, a sequence that includes registering the MIB subtrees that the subagent maintains and communicating the port number on which it listens.

Once SNMP starts, the following sequence occurs for each incoming SNMP request. This sequence is standard for SNMP implementations.

1. The master agent listens for incoming SNMP requests from clients on port 161. Authentication is limited to the validation of the community name. When a request arrives, the master agent communicates with the appropriate subagent.
2. Subagent routines collect the requested data and return the data to the master agent.
3. The master agent responds to the client from which the original request was made.

The SNMP shutdown procedure TCPIP$SNMP_SHUTDOWN.COM runs either from the general shutdown procedure TCPIP$SHUTDOWN.COM or can be run directly by the system manager.

TCPIP$SNMP_SHUTDOWN.COM does the following:

- Stops subagent processes and removes the SNMP images.
- Runs the SYS$STARTUP:TCPIP$SNMP_SYSHUTDOWN.COM procedure.

To ensure compatibility with previous versions, this procedure in turn runs SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$EXTENSION_MIB_SHUTDOWN.COM, which stops any additional subagent processes and deinstalls their images, if necessary.

13.1.2 Ensuring Access to Mounted Data

If the proxy setup between the SNMP server and the NFS server is not correct, the Host Resources MIB subagent cannot access data that has been mounted.

To ensure access to mounted data, set up a proxy to an anonymous user (for example, to TCPIP$NOBODY) on the NFS server system. For more information about adding proxy entries, see Chapter 20.

13.2 Managing the SNMP Service

The following command procedures are supplied to allow you to start up and shut down the SNMP service independently of TCP/IP Services:

- SYS$STARTUP:TCPIP$SNMP_STARTUP.COM allows you to start up the SNMP service.
- SYS$STARTUP:TCPIP$SNMP_SHUTDOWN.COM allows you to shut down the SNMP service.

Both the startup and shutdown procedures invoke the appropriate TCPIP$EXTENSION_MIB_*.COM file to ensure compatibility with previous versions of TCP/IP Services.

These files might be overwritten when you install subsequent versions of the TCP/IP Services product. For more information about these procedures, see Section 13.1.1.

To maintain site-specific SNMP logical names, commands, and parameter settings, you can create the following files:

- SYS$STARTUP:TCPIP$SNMP_SYSTARTUP.COM can be used as a repository site-specific definitions and parameters to be invoked when SNMP is started.
- SYS$STARTUP:TCPIP$SNMP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when SNMP is shut down.

Installing this version of TCP/IP Services will:

1. Purge existing TCPIP$SNMP_SYSTARTUP.COM and TCPIP$SYSHUTDOWN.COM files, leaving only the latest version.
2. Rename the remaining files to TCPIP$SNMP_SYSTARTUP.COM_OLD and TCPIP$SNMP_SYSHUTDOWN.COM_OLD.
3. Install new versions of TCPIP$SNMP_SYSTARTUP.COM and TCPIP$SYSHUTDOWN.COM files.

To preserve earlier versions of these files, copy them to another directory. New versions of TCP/IP Services will include instructions for merging the contents of the old versions with the current one.

13.3 Verifying the SNMP Installation

A separate installation verification procedure (IVP) exists for SNMP. To verify your configuration, complete these steps:

1. Log in to the SYSTEM account, or make sure that your process has the following privileges:
   - TMPMBX
   - NETMBX
   - SETPRV
2. Run the command procedure:

   ```
   $ @SYS$MANAGER:TCPIP$CONFIG
   ```

3. Enter option 7 (Run tests), and then option 2 from the Compaq TCP/IP Services for OpenVMS Test menu.

Note that, like the Internet IVP, the SNMP IVP requires that TCP/IP Services be running. (It does not require that SNMP be running.)

4. To run the SNMP IVP any time after exiting the configuration procedure, enter the following command:

```
$ RUN SYS$COMMON:[SYSTEST.TCPIP]TCPIP$SNMPIVP.EXE
```

13.3.1 SNMP Executable and Command Files

Table 13–2 lists the names of the primary SNMP executable and command files and their locations. For a list of files that help you build your own subagent, see the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

Table 13–2 SNMP Executable, Command, and Data Files

File	Location	Function
TCPIP$ESNMP_SERVER.EXE	SYS$SYSTEM	Master agent image.
TCPIP$OS_MIBS.EXE	SYS$SYSTEM	MIB II subagent image.
TCPIP$HR_MIB.EXE	SYS$SYSTEM	Host Resources MIB subagent image.
TCPIP$SNMP_REQUEST.EXE	SYS$SYSTEM	Simple MIB browser.
TCPIP$SNMP_TRAPSND.EXE	SYS$SYSTEM	Program for sending trap messages.
TCPIP$SNMP_TRAPRCV.EXE	SYS$SYSTEM	Program for receiving trap messages.
TCPIP$ESNMP_SHR.EXE	SYS$SHARE	Routines in the eSNMP application programming interface (API).
TCPIP$SNMP_STARTUP.COM	SYS$STARTUP	Installs master and subagent images and runs TCPIP$SNMP_RUN.COM.
TCPIP$SNMP_RUN.COM	TCPIP$SYSTEM	Starts the master agent and subagents.
TCPIP$SNMP_SHUTDOWN.COM	SYS$STARTUP	Stops the master agent and subagents.
TCPIP$SNMP_SYSTARTUP.COM	SYS$STARTUP	Sets site-specific configuration values on startup.
TCPIP$SNMP_SYSHUTDOWN.COM	SYS$STARTUP	Sets site-specific configuration values on shutdown.
TCPIP$EXTENSION_MIB_STARTUP.COM	SYS$SYSDEVICE:[TCPIP$SNMP]	Starts custom subagents.
TCPIP$EXTENSION_MIB_SHUTDOWN.COM	SYS$SYSDEVICE:[TCPIP$SNMP]	Shuts down custom subagents.
TCPIP$VMS_SNMP_CONF.DAT	SYS$SYSDEVICE:[TCPIP$SNMP]	User-editable configuration data file.

(continued on next page)

Table 13–2 (Cont.) SNMP Executable, Command, and Data Files

File	Location	Function
TCPIP$SNMP_CONF.DAT	SYS$SYSDEVICE:[TCPIP$SNMP]	Configuration data file used in the startup of the master agent and standard subagents.

13.4 Configuring SNMP

You can configure SNMP in three ways, which may be used in combination:

- Using the standard TCPIP$CONFIG.COM procedure and the SET CONFIGURATION SNMP command. These methods write configuration information into the TCP/IP Services configuration database file TCPIP$CONFIGURATION.DAT. Section 13.4.1 describes how to use TCPIP$CONFIG to initially configure SNMP.
- Editing the text configuration file TCPIP$VMS_SNMP_CONF.DAT, located in the SYS$SYSDEVICE:[TCPIP$SNMP] directory. This method provides options not available with TCPIP$CONFIG and the SET CONFIGURATION SNMP command.

  Note

  Although the OpenVMS SNMP configuration file is based on the UNIX implementation, there are several important differences. For example, the option `snmpEnableAuthenTraps` is not used. See the description of specific options for details.

  The configuration file is described in Section 13.4.3.
- Assigning logical names. This method provides the same options as the text configuration file. For more information, see Section 13.4.3.

If the same option is defined in multiple ways, the configuration methods are resolved as follows:

- Values specified through TCPIP$CONFIG or SET CONFIGURATION SNMP take precedence over any options specified in the TCPIP$VMS_SNMP_CONF.DAT file or set with logical names.
- Values specified in the TCPIP$VMS_SNMP_CONF.DAT file take precedence over logical name settings.

13.4.1 Initial SNMP Configuration

SNMP runs as a TCP/IP service. To be sure all SNMP-related files are included and enabled properly, run the TCPIP$CONFIG configuration procedure to configure SNMP initially or to set up a new configuration. When you enable SNMP during TCPIP$CONFIG, the procedure prompts you for the correct parameters.

Note

You cannot use TCPIP$CONFIG to modify your existing SNMP configuration; TCPIP$CONFIG is intended only to set up a new SNMP configuration.

To modify the current SNMP configuration (for example, to specify an additional community name and address), you must enter the SET CONFIGURATION SNMP command with applicable qualifiers.

When you run TCPIP$CONFIG after a TCP/IP Services upgrade, be sure to disable and then reenable the SNMP service.

You supply the following information about your host when you configure SNMP initially during TCPIP$CONFIG or when you issue the SET CONFIGURATION SNMP command to modify your existing SNMP configuration. For detailed information about the SET CONFIGURATION SNMP command and qualifiers, see the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.

- The name of the person to contact about the system. For example:

  ```
  TCPIP> SET CONFIGURATION SNMP/CONTACT="Sam Spade"
  ```

- The physical location of the system. For example:

  ```
  TCPIP> SET CONFIGURATION SNMP -
  _TCPIP> /LOCATION=(FIRST="Falcon Building",SECOND="Los Angeles, CA")
  ```

- The community information used to authenticate requests from a network manager and to determine the addresses to which trap messages are sent.

  SNMP network management clients are grouped into communities as specified in RFC 1157. You can define one or more communities, which your master agent uses to authenticate requests.

  The parameters you specify for each community are as follows:

  – Community name

    The name associated with the community. The standard community is "public." You can choose to not provide this community name when you run TCPIP$CONFIG. Answer No to the question "Do you want to provide the public community." If you disable the public community, you may need to reconfigure SNMP clients in your environment.

    Community names are case sensitive. When you use TCPIP$CONFIG to specify a community name, do not use quotation marks to preserve the case. The case is preserved exactly as you enter it. However, if you customize your existing SNMP configuration using the SET CONFIGURATION SNMP command, make sure you enclose the community name in quotation marks to preserve the case. If you do not enclose the community name in quotation marks, the name is changed to all uppercase.

    The community name must be a string of alphanumeric characters. You cannot include a space or other nonalphanumeric character in the community name.

    You can also modify the community name using the `community` option in the configuration file, as described in Table 13–4.

– Community address

  The address associated with the community. One community name can have multiple addresses in its entry. For example:

```
TCPIP> SET CONFIGURATION SNMP /ADDRESS=(6.10.1.2,100.2.2.1)
```

  Specifying address 0.0.0.0 for READ and WRITE allows any host the type of access specified. To allow any network manager to monitor your system remotely, specify the standard community name (public, in lowercase letters) with address 0.0.0.0. For example:

```
TCPIP> SET CONFIGURATION SNMP /COMMUNITY="public" /ADDRESS=0.0.0.0
```

  Traps are sent to UDP port 162 on hosts for all trap addresses regardless of community name. The use of address 0.0.0.0 on a trap means that traps are not sent unless another address is also specified.

– Types of access

  The types of access associated with the community are described in the following table:

Access Type	Allows the Master Agent and Subagent to...
READ	Respond to a client's read requests (gets) for network information. Default. Members of a read-only community do not have write access to the SNMP MIB objects.
TRAP	Send alert messages (traps) to a client as a result of unusual events. For example, a trap message is sent to the client as a result of a get request that specifies an unauthorized community string (authenticationFailure).
WRITE	Process client write requests (sets) on your host's MIB data items.

For example, to allow the master agent to respond to client get requests, enter:

```
TCPIP> SET CONFIGURATION SNMP /COMMUNITY="public" /TYPE=READ
```

To configure your host to allow client set requests, use the /FLAGS=SETS qualifier. For example:

```
TCPIP> SET CONFIGURATION SNMP /COMMUNITY="public" /FLAGS=SETS
```

13.4.2 Displaying the Current SNMP Configuration

To display configuration information in the SNMP configuration database, use the SHOW CONFIGURATION SNMP command. Use the /FULL qualifier if you want to display the addresses that the agent recognizes as members of the community. For example, enter:

```
TCPIP> SHOW CONFIGURATION SNMP /FULL

SNMP Configuration

Flags:    AuthenTraps  Sets

Contact:  Sam Spade

Location
  First:  Falcon Building
  Second: Los Angeles, CA

Community           Type       address_list

public              Read       0.0.0.0
```

```
writeit             Read Write 9.20.208.53
trapit              Read Trap  9.20.208.53, 9.20.208.100
```

In this example, the configuration allows read access to any client on any host through the "public" community and read/write access to the client on host 9.20.208.53 through the "writeit" community. In addition, trap messages are sent to UDP port 162 on hosts 9.20.208.53 and 9.20.208.100.

Alternatively, you can display the configuration options in the SNMP configuration text file described in Section 13.4.3. See Section 13.6.5.2 for more information.

13.4.3 SNMP Options

You can configure the way SNMP runs by entering SNMP options into the SNMP configuration file TCPIP$VMS_SNMP_CONF.DAT.

When it starts, the SNMP master agent creates the temporary file SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$TMP_SNMP_CONF.DAT from data in the standard TCP/IP configuration database file TCPIP$CONFIGURATION.DAT. A few versions of this file are preserved in case they are needed for troubleshooting. The master agent appends this temporary file to TCPIP$VMS_SNMP_CONF.DAT to produce the master configuration file TCPIP$SNMP_CONF.DAT.

When the standard OS_MIBS and HR_MIB subagents start up, they read TCPIP$SNMP_CONF.DAT. Only the master agent and these standard subagents use values in the text files.

By default, custom subagents do not take advantage of the configuration options. To take advantage of these options, you must assign a logical that is visible to the subagent process. The following example shows how to define TCPIP$SNMP_GEN_LOGFILE logical to set the snmp_gen_logfile configuration option:

```
$ ASSIGN/SYSTEM 1 TCPIP$SNMP_GEN_LOGFILE
```

If a configuration option is not handled by the eSNMP API, the subagent must include an explicit genenv( ) or similar call to access the value of the option.

13.4.3.1 Using Logical Names to Configure SNMP

Most configuration options have a corresponding logical name. In some cases, you can define system logical names as an alternative to entering a value in the text file. See Section 13.4.3.4 for a list of the options and their associated logical names.

13.4.3.2 Dynamic Options

Some options are available for you to change dynamically; that is, without shutting down and restarting the SNMP service. To change configuration values dynamically, you can do one of the following:

- Define the appropriate logical name.
- Edit the configuration file, then define snmp_signal to be sighup. Be sure to deassign snmp_signal afterwards to prevent continuous rereading of the configuration file.

13.4.3.3 Modifying the Configuration File

The master agent and the subagents convert lines in the configuration file that begin with the OpenVMS-specific `config` command to user-mode process logicals by adding the prefix TCPIP$. For example, SNMP_GEN_LOGFILE becomes TCPIP$SNMP_GEN_LOGFILE. (This mechanism does not apply to options with other keywords, such as `trap`.) Because the logicals are local to agent processes, they are not visible to a DCL command SHOW LOGICAL issued in another process.

If there are lines with duplicate configuration tags, the last line supersedes all others. Because the temporary file TCPIP$TMP_ CONF.DAT (described in Section 13.4.3) is appended after the user-editable TCPIP$VMS_SNMP_CONF.DAT file, the standard TCPIP configuration values from that temporary file always supersede those from the user-edited file.

Lines in the configuration file that begin with a pound sign (#) are ignored. The pound sign is the comment character.

Option names and values are not case sensitive. Boolean values are considered on if the option is present with no value. Otherwise, they are considered off. Thus, to turn off an option that was enabled at startup, you must specify zero as the value.

If you specify a value that is longer than the limit, the value is converted to hexadecimal and then truncated. For example, if you specify the value 257 in place of an 8-bit unsigned value, it is converted to hexadecimal (0101) and truncated to 1.

13.4.3.4 SNMP Configuration Options

Most of the SNMP options set in the TCPIP$VMS_SNMP_CONF.DAT file must be entered using the following syntax:

`config` *`option-name value`*

There are several types of SNMP configuration options:

- Logging options, described in Table 13–3. These options control the way messages are logged.
- Operation options, described in Table 13–4 These options control the operational settings for SNMP. Some of these options cannot be set by using a logical name.
- Timing options, described in Table 13–5. These options control the way timeouts are handled.
- Testing and troubleshooting options, described in Table 13–6. These options are useful when you are testing SNMP functions and troubleshooting subagent problems.
- Backward-compatibility options, described in Table 13–7. These options are available to provide compatibility with subagents developed under previous versions of SNMP.

Except for the community name, option values are not case sensitive.

Table 13–3 SNMP Logging Options

SNMP_GEN_LOGFILE	
Logical name:	TCPIP$SNMP_GEN_LOGFILE
Format:	`config SNMP_GEN_LOGFILE 1`
Description:	Redirects messages to SYS$OUTPUT and records them in the following files: • TCPIP$ESNMP_SERVER*process-id*.LOG, where *process-id* is the eight-digit hexadecimal process identifier of the master agent. • TCPIP$ESNMP_RESIDENT_SUBAGENT*process-id*.LOG, where *process-id* is the eight-digit hexadecimal process identifier of the resident subagent. • TCPIP$OS_MIBS*process-id*.LOG, where *process-id* is the eight-digit hexadecimal process identifier of the MIB II subagent. • TCPIP$HR_MIB*process-id*.LOG, where *process-id* is the eight-digit hexadecimal process identifier of the Host Resources MIB subagent.
Type:	Dynamic
SNMP_SUPPRESS_LOGGING_TIMESTAMP	
Logical name:	SNMP_SUPPRESS_LOGGING_TIMESTAMP
Format:	`config TCPIP$SNMP_SUPPRESS_LOGGING_TIMESTAMP 1`
Description:	Specifies whether a timestamp is included in the log message. If not defined, a timestamp is included. The value can be 1 (to prevent timestamp information from being included) or 0 (to allow timestamp information to be included; the default).
Type:	Dynamic
SNMP_VERBOSE_LOGGING	
Logical name:	TCPIP$SNMP_VERBOSE_LOGGING
Format:	`config SNMP_VERBOSE_LOGGING 1`
Description:	Specifies whether to log detailed information or not. The value can be 1 (to log detailed information) or 0 (to log the default amount of information).
Type:	Dynamic

Table 13–4 SNMP Operation Options

COMMUNITY	
Logical name:	Not available
Format:	`COMMUNITY` *`name address type`*
Description:	Specifies the community name. See Section 13.4 for more information about specifying a community name.
Type:	Dynamic
SNMPENABLEAUTHENTRAPS	
Logical name:	Not available
Format:	`SNMPENABLEAUTHENTRAPS`
Description:	This configuration option reflects the setting of the /FLAGS=AUTHENTICATION qualifier to the SET CONFIGURATION SNMP command and is included in the configuration file for backward compatibility. This option in the configuration file is ignored.
Type:	Not dynamic
SNMP_RESTARTS	
Logical name:	TCPIP$SNMP_RESTARTS
Format:	`config SNMP_RESTARTS 5`
Description:	Specifies the maximum number of times to restart a subagent. The default for OS_MIBS and HR_MIB is 3.
Type:	Not dynamic
SNMP_SELECT_ERROR_LIMIT	
Logical name:	TCPIP$SNMP_SELECT_ERROR_LIMIT
Format:	`config SNMP_SELECT_ERROR_LIMIT 500`
Description:	Specifies the number of iterations for the error limit. The default value is 100.
Type:	Not dynamic

(continued on next page)

Table 13–4 (Cont.) SNMP Operation Options

SNMP_SIGNAL	
Logical name:	TCPIP$SNMP_SIGNAL
Format:	`DEFINE TCPIP$SNMP_SIGNAL` *`value`*
Description:	Simulates a UNIX-style signal that affects the way agents operate. Following is a list of values: – SIGUSR1—causes a dump of MIB registration area with contexts to the following log file: `SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$SNMP_DUMP.LOG` – SIGHUP—rereads the configuration file. – SIGINT—causes the process to exit. – SIGTERM—same as SIGINT. – SIGUSR2—turns on tracing. – SIGCHLD—turns off tracing. Do not set this option in the configuration text file. After setting the logical name, be sure to reset it to prevent system performance problems.
Type:	Dynamic
SYSNAME	
Logical name:	Not available
Format:	`SYSNAME` *`host-name`*
Description:	Specifies the SNMP host name. This host name is used only by SNMP. You can reset the host name by editing this option and then restarting the master agent.
Type:	Not dynamic
SYSCONTACT	
Logical name:	Not available
Format:	`SYSCONTACT` *`contact-information`*
Description:	Specifies the contact information. Do not modify this option. Use TCPIP$CONFIG or the SET CONFIGURATION SNMP command to change the information associated with this option.
Type:	Not dynamic

(continued on next page)

Table 13–4 (Cont.) SNMP Operation Options

SYSLOCATION	
Logical name:	Not available
Format:	SYSLOCATION *host-location*
Description:	Specifies the host or contact location information. Do not modify this option. Use TCPIP$CONFIG or the SET CONFIGURATION SNMP command to change the information associated with this option.
Type:	Not dynamic

trap	
Logical name:	Not available
Format:	trap *trap-name version IP-address*
Description:	Specifies: – The name of the trap (*trap-name*). – Whether to trap for SNMP Version 1 requests only (*version*). Specify V1 for Version 1 traps only. Specify V2C for both Version 1 and Version 2 traps. – The internet address of the client (*address*). Do not specify 0.0.0.0 for the client address. For information about setting individual trap types depending on the destination host, see Section 13.6.5.3.
Type:	Not dynamic

Table 13–5 Timing and Timeout Handling Options

AGENTX_SESSION_TIMEOUT	
Logical name:	TCPIP$AGENTX_SESSION_TIMEOUT
Format:	config AGENTX_SESSION_TIMEOUT *seconds*

(continued on next page)

Table 13–5 (Cont.) Timing and Timeout Handling Options

AGENTX_SESSION_TIMEOUT

Description: Specifies the default timeout for a session between a subagent and the master agent. Subagents can supersede this value when they register their MIBs.

The value of this option is used by both the master agent and the subagent. Normally, all subagents running on the same host have the same timeout value, which is specified by this option.

When the subagent reads the value of this option, the value is interpreted as follows:

- If the option is not defined, the default value of 3 seconds is assumed.
- If the option is set to 0, the timeout value used by the master agent is used.
- If the option is set to a nonzero integer, that value is used instead of the master agent's default timeout value.

When the master agent reads the value of this option, the value is interpreted as follows:

- If the option is not defined, the default value of 3 seconds is assumed.
- If the option is set to a value greater than 0, this timeout value is used, unless a different value has been specified for the subagent.
- Do not set the value of this option to 0.

The maximum value you can specify is 255. This option can be used to increase the timeout for communication between the master agent and subagents on a slow system.

Type: Dynamic

SNMP_MASTER_TIMEOUT

Logical name: TCPIP$SNMP_MASTER_TIMEOUT

Format: `config SNMP_MASTER_TIMEOUT` *`seconds`*

Description: Specifies (in seconds) the default time to wait listening for an SNMP request. The default is 10 seconds.

Type: Not dynamic

(continued on next page)

Table 13–5 (Cont.) Timing and Timeout Handling Options

SNMP_ARE_YOU_THERE_TIME	
Logical name:	TCPIP$SNMP_ARE_YOU_THERE_TIME
Format:	`config SNMP_ARE_YOU_THERE_TIME` *seconds*
Description:	Specifies the time subagents wait between sending the `esnmp_are_you_there( )` message to the master agent. For the OS_MIBS and the HR_MIB, the default is 5400 seconds (90 minutes). If you also specify the `SNMP_INACT_TIME` option, make sure the value of the SNMP_ARE_YOU_THERE_TIME option is less than or equal to the value of the SNMP_INACT_TIME option.
Type:	Dynamic

SNMP_POLL_TIME	
Logical name:	TCPIP$SNMP_POLL_TIME
Format:	`config SNMP_POLL_TIME` *seconds*
Description:	Specifies the interval between times that interface counts and other values are reset for standard subagents.
Type:	Dynamic

SNMP_INACT_TERM	
Logical name:	TCPIP$SNMP_INACT_TERM
Format:	`config SNMP_INACT_TERM` *n*
Description:	In this format, *n* can be 1 (to terminate the master agent) or 0 (to never terminate the master agent). Specify the amount of time to wait using the SNMP_INACT_TIME option.
Type:	Dynamic

SNMP_INACT_TIME	
Logical name:	TCPIP$SNMP_INACT_TIME
Format:	`config SNMP_INACT_TIME` *seconds*
Description:	Specifies (in seconds) the amount of time that must pass before the subagent is considered inactive (that is, the amount of time during which the master agent receives no message from the subagent). See also the SNMP_INACT_TERM and SNMP_ARE_YOU_THERE_TIME options.
Type:	Dynamic

Time-related parameters are important in determining the responsiveness of the SNMP agents to client requests, particularly on systems with limited memory or those that are heavily loaded.

On startup, each subagent first sets up a default session timeout (see the AGENTX_SESSION_TIMEOUT option). It then registers its MIB regions. The subagent can register each of its MIB regions with a different timeout. A value of 0 causes the session timeout for the entire subagent to be used.

The master agent listens for SNMP requests. The timeout value is 10 seconds, unless the SNMP_MASTER_TIMEOUT option has been defined. After a timeout occurs, the master agent updates counters, checks for requests, then loops to wait for another SNMP request. When an SNMP request arrives, the master agent determines which if any registered subagents can handle it. It then resets the SNMP_MASTER_TIMEOUT timeout to use the maximum of the timeouts for all MIB regions involved.

When it is not processing an SNMP request, a subagent may send are_you_there messages to the master agent at a default interval determined by the subagent. For the chess example, the default is 30 seconds; for the OS_MIBS and HR_MIB subagents, the default is 5400 seconds (90 minutes). Both values are derived from those used in the UNIX implementation of SNMP; the second value was set high to minimize system overhead.

The following relationships among configuration option values are recommended but are not enforced. See the descriptions of the specific options for details.

- SNMP_ARE_YOU_THERE_TIME and SNMP_INACT_TIME

  The SNMP_ARE_YOU_THERE_TIME option determines the time between are_you_there messages. If the SNMP_INACT_TERM option is set, and if the master agent does not receive any SNMP request or are_you_there mesages from a subagent during the time associated with the SNMP_INACT_TIME option, the master agent automatically exits. By default, the SNMP_INACT_TERM option is not set.

  If the SNMP_ARE_YOU_THERE_TIME option is not set and no external SNMP requests are received, the master agent will exit even if subagents are still active.

- SNMP_INACT_TIME and SNMP_POLL_TIME

  The values for these options should be a multiple of the value of the SNMP_MASTER_TIMEOUT option.

  The master agent checks whether these intervals have elapsed after the time specified by the SNMP_MASTER_TIMEOUT option. Therefore, a value for these two options that is not a multiple of SNMP_MASTER_TIMEOUT will have the same effect as one that is the next higher multiple.

- The client should allow a large enough timeout interval to accommodate the server to avoid query failures or unnecessary retries. Particular care is required when network load is high and when communicating with heavily used servers and those in which tracing is turned on. See Table 13–6 for details on using trace.

Table 13–6 Testing and Troubleshooting Options

ACCEPT	
Logical name:	Not available
Format:	`accept` *IP-address*
Description:	If nonlocal subagents are allowed (using the SNMP_ALLOW_INET_TRANSPORT, AGENT_INET_ADDR, or AGENTX_INET_PORT option), the ACCEPT option specifies the IP address of the host from which a connection will be accepted. If these options are not set, connections from nonlocal subagents are rejected. To allow access from all subagents, specify the *IP-address* as 0.0.0.0.
Type:	Dynamic
AGENTX_LOCAL_PORT	
Logical name:	TCPIP$AGENTX_LOCAL_PORT
Format:	`config AGENTX_LOCAL_PORT` *port number*
Description:	Specifies the local port number from which to accept nonlocal subagent connections.
Type:	Dynamic
AGENTX_INET_PORT	
Logical name:	TCPIP$AGENTX_INET_PORT
Format:	`config AGENTX_INET_PORT` *port number*
Description:	Specifies the TCP/IP port number from which to accept connections from nonlocal subagents.
Type:	Dynamic
SNMP_ALLOW_INET_TRANSPORT	
Logical name:	TCPIP$SNMP_ALLOW_INET_TRANSPORT
Format:	`config SNMP_ALLOW_INET_TRANSPORT` *n*
Description:	Specifies whether the master agent accepts connections from nonlocal subagents.
Type:	Dynamic

(continued on next page)

Table 13–6 (Cont.) Testing and Troubleshooting Options

SNMP_TRACE	
Logical name:	TCPIP$SNMP_TRACE
Format:	`config TCPIP$SNMP_TRACE` *n*
Description:	Allows you to direct trace log messages to standard log files when agents are running in normal production mode. (Alternatively, you can get trace logs while running the subagent in interactive mode, as described in Section 13.6.4.) Running with tracing produces a great deal of output and may slow down the system. In addition, utilities like the MIB browser (`snmp_request`) may need a longer timeout interval when running with tracing on. The type of data and the amount of data logged for custom subagents depends on how the subagents are programmed, except for the logging that is handled automatically by the eSNMP API routines. The chess example code provides some samples of using the ESNMP_LOG macro.
Type:	Not dynamic

Table 13–7 Backward-Compatibility Options

SNMP_PROHIBIT_DUPLICATE_REGISTRATIONS	
Logical name:	TCPIP$SNMP_PROHIBIT_DUPLICATE_REGISTRATIONS
Format:	`config SNMP_PROHIBIT_DUPLICATE_REGISTRATIONS` *n*
Description:	In this format, *n* can be 1 (to set the option), or 0 (to turn the option off). If this option is set, a subagent that tries to register with the same name as a previously registered subagent will be rejected. By default, duplicate registrations are allowed; the AgentX protocol does not check for duplicate subagents based on the subagent name.
Type:	Dynamic

SNMP_V1_TRAP_DEFAULT	
Logical name:	TCPIP$SNMP_V1_TRAP_DEFAULT
Format:	`config SNMP_V1_TRAP_DEFAULT` *n*
Description:	In this format, *n* can be 1 (to set the option), or 0 (to turn the option off). When this option is set, traps defined in the TCPIP$CONFIG.COM procedure or using the TCP/IP management command SET CONFIGURATION SNMP are sent in SNMP Version 1 format. The default is to send these types of traps in Version 2 format.
Type:	Dynamic

13.5 SNMP Log Files

Unless the SNMP_TRACE option is set, output from the SNMP master agent and subagent processes to SYS$OUTPUT is redirected to the following files:

- TCPIP$SNMP_RUN.LOG
- TCPIP$OS_MIBS.LOG
- TCPIP$HR_MIB.LOG

The output is written to these files continuously while SNMP processes are running. Buffering may cause a delay in writing to disk; therefore, if a process is terminated abnormally, some data may be lost.

While processes are running, output for SYS$ERROR can be redirected to other files. See Section 13.4.3 for information about controlling this. In addition, the master agent and subagents may write to SYS$ERROR. This output is redirected to the following files:

- TCPIP$SNMP_RUN.LOG
- TCPIP$OS_MIBS.ERR
- TCPIP$HR_MIB.ERR

Unlike a regular log or a trace log, this output is written when the corresponding SNMP process terminates. Therefore, abnormal termination can cause data to be lost.

All of the listed log files are located in the SYS$SYSDEVICE:[TCPIP$SNMP] directory. The configuration-related files described in Section 13.4.3 are also stored there. TCP/IP Services does not allow you write to log files in other directories.

The log level and specific events during processing determine how much information is recorded in the log files; log files can be empty or nonexistent.

The log files contain startup and event information and additional messages, depending on the logging level specified for an agent. The SNMP logging facility uses three logging levels:

- Trace (logs trace, warning, and error messages)
- Warning (logs warning and error messages)
- Error

The default logging level for the master agent and standard subagents is Warning. Because the Chess example subagent does not use a default, messages are captured only if you specify tracing, as described in Section 13.6.4.

Many logging options are configurable using the text configuration file SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$VMS_SNMP_CONF.DAT; see Table 13–3 for more details.

The following log files exist under normal production conditions if special configuration options are not used. In most cases, a new version of each file is created each time SNMP is started:

Agent	Process	SYS$OUTPUT	SYS$ERROR
Master agent	TCPIP$SNMP	TCPIP$SNMP_ RUN.LOG	TCPIP$SNMP_RUN.LOG
Resident subagent	TCPIP$SNMP	TCPIP$SNMP_ RUN.LOG	TCPIP$SNMP_RUN.LOG
OS_MIBS[1]	TCPIP$OS_MIBS	TCPIP$OS_MIBS.LOG	TCPIP$OS_MIBS.ERR
HR_MIB	TCPIP$HR_MIB[1]	TCPIP$HR_MIB.LOG	TCPIP$HR_MIB.ERR

[1]If no output has been generated, a .LOG or .ERR file might not exist.

If the configuration option SNMP_GEN_LOGFILE is set, files in the preceding table continue to be used for SYS$ERROR data. For SYS$OUTPUT data, as soon as the agents detect the option, data is written to the following files, where *process-ID* is the hexadecimal process ID of the process listed:

Agent	Process	SYS$OUTPUT
Master agent	TCPIP$SNMP	TCPIP$ESNMP_SERVER*process-ID*.LOG
Resident subagent	TCPIP$SNMP	TCPIP$ESNMP_RESIDENT_SUBAGENT*process-ID*.LOG
OS_MIBS	TCPIP$OS_MIBS	TCPIP$OS_MIBS*process-ID*.LOG
HR_MIB	TCPIP$HR_MIB	TCPIP$HR_MIB*process-ID*.LOG

Unless it is suppressed, the timestamp gives a line-by-line record of when output was written to each file and is useful in resolving timing-related problems.

The SNMP_GEN_LOGFILE option does not affect the name of the output file for customer written subagents. Customer-written subagents generate files based on the IMAGENAME symbol in SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$EXTENSION_MIB_RUN.COM.

For details about logging from customer extension subagents, refer to the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

13.6 Solving SNMP Problems

The following sections contain information about how to analyze and solve many SNMP problems. Be sure to configure SNMP according to the instructions in this guide, and use the information here and in the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide when writing your own subagents.

13.6.1 Multiple SNMP Processes Displayed for SHOW SYSTEM Command

When you enter the DCL command SHOW SYSTEM during the TCPIP or SNMP startup sequence, the process TCPIP$SNMP_*n* may appear in the display without the subagent processes (TCPIP$OS_MIBS and TCPIP$HR_MIB). This is because TCPIP$SNMP is the main SNMP process started by the TCP/IP kernel when the SNMP service is enabled; it starts the subagents as detached processes, and then continues to run as the master agent. The number at the end of this process name reflects the number of times this main process has started since SNMP has been enabled.

13.6.2 Problems Starting and Stopping SNMP Processes

If there are startup errors noted in the SNMP log files, or if SNMP startup seems normal but one or more of the SNMP processes disappears, follow these steps:

1. Check the log files for any errors indicating timeouts, protection problems, or configuration errors.
2. Start up the master agent and subagents by running the images interactively and enabling tracing (see Section 13.6.4).

To verify the SNMP installation, enter the command SHOW CONFIGURATION SNMP, as described in Section 13.4.2.

To stop all SNMP processes, enter:

```
$ @SYS$STARTUP:TCPIP$SNMP_SHUTDOWN
```

If you disable the SNMP service by entering the DISABLE SERVICE SNMP command, automatic restarts are prevented, but detached SNMP master and subagent processes are not stopped.

13.6.3 Restarting MIB Subagent Processes

Usually the SNMP master agent and subagent processes start up and are shut down together as described in Section 13.1.1.

If the SNMP master agent process stops for any reason, TCP/IP Services attempts to restart it and, if successful, increments the count (*n*) in the process name TCPIP$SNMP_*n*. As part of the startup sequence, any subagents that have stopped will be restarted. If a subagent process has not stopped, an attempt to restart it will have no effect because OpenVMS does not allow a duplicate process name (unlike the SNMP master agent, subagent names do not include a startup count).

If the master agent continues to run but a subagent stops, there is no automatic restart attempt. You can correct the problem by doing one of the following:

- Restart TCP/IP Services.
- Restart SNMP.
- Manually stop the TCPIP$SNMP_*n* process to force a master agent restart.
- Configure the SNMP variable AUTRESTARTS and stop all the subagent processes. See Section 13.4.3 for more information.

13.6.4 Obtaining Trace Log Messages

To get trace log messages you can:

- Configure SNMP to enable trace output while SNMP continues processing.
- Enable tracing while running SNMP interactively.

To configure SNMP to log tracing messages while it is running, set the `snmp_trace` configuration option. With this option enabled, trace output is produced and written to standard logs (see Section 13.5) when agents are run in normal production mode.

See Section 13.4.3 for details about the configuration options and about how to enable those options dynamically or without running interactively.

To obtain trace log messages interactively, follow these steps:

1. Shut down SNMP. Enter:

   ```
   $ @SYS$STARTUP:TCPIP$SNMP_SHUTDOWN
   ```

2. From separate windows, run the master agent and subagents interactively. For example, run each image by entering the following commands in separate windows:

   ```
   $ MCR TCPIP$ESNMP_SERVER -T
   $ MCR TCPIP$OS_MIBS -TRACE
   $ MCR TCPIP$HR_MIB -TRACE
   ```

   To specify custom subagents located in directories other than SYS$SYSTEM, use the MCR command and specify the full directory path. For example, to run the Chess example subagent with trace logging, enter the following command:

   ```
   $ MCR SYS$COMMON:[SYSHLP.EXAMPLES.TCPIP.SNMP]TCPIP$CHESS_SUBAGENT -TRACE
   ```

When agents are run interactively, output comes to the terminal unless the SNMP_GEN_LOGFILE option is enabled.

Running in trace mode can produce a great deal of output, and also slow down performance significantly. Programs like browsers may need to allow a longer timeout interval under these circumstance. For example, use the `-W` with the supplied MIB browser.

For more information about the MIB browser supplied with TCP/IP Services, and on using tracing with custom subagents, see the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

The type of trace data written depends on the way the subagent routines are programmed, except for logging handled within eSNMP API routines. For more details, see the Chess example code.

13.6.5 Processing Set Requests and Traps

To make sure that the master agent processes SNMP `Set` requests from management clients correctly, follow these steps:

1. Configure SNMP to allow the master agent to process `Set` requests, either by using the TCPIP$CONFIG.COM configuration procedure or by using the SET CONFIGURATION SNMP command.
2. Make sure that the management client is configured correctly for `Get` and `Set` requests, as described in the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.
3. Configure write communities as needed on the OpenVMS server. Refer to Section 13.6.5.2.2 for more information.
4. Make sure that the requested MIB variable is defined with write access and implemented as such in the subagent. Refer to the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide for more information.

If SNMP is not responding to `Set` requests after you follow these steps, refer to Section 13.6.6 for troubleshooting procedures and Section 13.6.5.2.2 to check the community configuration information.

13.6.5.1 Enabling Set Request Processing and Authentication Traps

On an OpenVMS server, configure SNMP with the /FLAGS=SETS qualifier to the management command SET CONFIGURATION SNMP, or enable SNMP during the configuration procedure (TCPIP$CONFIG) by answering Yes to the question `Do you want to allow clients modify (SET) access?`

To enable `set` requests and traps on an existing SNMP configuration, enter the SET CONFIGURATION SNMP command with the /FLAGS=*options* qualifier, specifying the SETS option to enable `set` requests and the AUTHEN_TRAPS option to enable sending authentication failure traps.

When you enter the SET CONFIGURATION SNMP command and qualifiers, take the following information into consideration:

- SNMP functions without the need to configure flags for `set` commands (/FLAGS=SETS) and authentication traps (/FLAGS=AUTHEN_TRAPS). Note that when you enter the SHOW CONFIGURATION SNMP command, the keywords associated with these flags are displayed as follows:

  ```
  Flags:    AuthenTraps Sets
  ```

- The /FLAGS=SETS qualifier is required to enable SNMP client `set` command requests. If `set` commands are not enabled, the client receives a "`no such variable`" message, even if access type requirements are met. (See the command guidelines in Section 13.6.5.1.)

- The /FLAGS=AUTHEN_TRAPS qualifier allows the SNMP master to send trap messages to specified trap community addresses when MIB access with a community name is not supported by the agent. This also allows the master to send trap messages when the agent does not grant the host the access required for a request (for example, READ for a `get` request or WRITE for a `set` request).

For example, to enable response to `set` requests and to allow authentication traps on an existing SNMP configuration, enter the following command:

```
TCPIP> SET CONFIGURATION SNMP/FLAGS=(SETS,AUTHEN_TRAPS)
```

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* guide for detailed information about the SET CONFIGURATION SNMP command.

Restart SNMP after making any changes to the configuration.

13.6.5.2 Displaying Configuration Information

When you enter the SHOW CONFIGURATION SNMP command to display your current SNMP configuration, the information associated with the /FLAGS=*options* qualifier is displayed as follows:

```
Flags:    AuthenTraps Sets
```

SNMP will function even if you do not include the /FLAGS=SETS and /FLAGS=AUTHEN_TRAPS qualifiers.

To remove flags that were set previously, enter the following commands:

```
TCPIP> SET CONFIGURATION /FLAGS=NOSETS
```

```
TCPIP> SET CONFIGURATION /FLAGS=NOAUTHEN_TRAPS
```

Alternatively, you can display configuration information in the SNMP configuration file (SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$VMS_SNMP_CONF.DAT). The configuration file displays more information than the SHOW CONFIGURATION SNMP command when multiple types of traps or addresses for them have been defined. For example:

```
$ TYPE SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$VMS_SNMP_CONF.DAT

trap      V1 elmginkgo 15.9.0.200
community     alternate 15.4.3.2 read
community         public  0.0.0.0 read
community         TRAPIT 1.2.4.5 write
trap      v2c     TRAPIT 1.2.4.5
community         rw     10.1.1.3       write
community         rw     15.9.0.200     write
```

Note that the first two lines of the configuration file are not displayed by the following SHOW CONFIGURATION SNMP/FULL command:

```
TCPIP> SHOW CONFIGURATION SNMP/FULL

Community           Type           Address_list

public              Read           0.0.0.0

TRAPIT              Read Write Trap
                                   1.2.4.5

rw                  Read Write     10.1.1.3,  15.9.0.200
```

13.6.5.2.1 Specifying Location and Contact Information To specify the location and contact information, include the /LOCATION and /CONTACT qualifiers on the SET CONFIGURATION SNMP command line.

If you do not specify the location and contact information, it is displayed as "not defined" by the SHOW CONFIGURATION SNMP/FULL command. For example:

```
TCPIP> SHOW CONFIGURATION SNMP/FULL

SNMP Configuration

Flags:    Sets

Contact:  not defined

Location: not defined
```

To remove a previously specified location, enter:

```
TCPIP> SET CONFIGURATION SNMP /LOCATION=(NOFIRST,NOSECOND)
```

Note

If you enabled SNMP when you had a previous version of TCP/IP Services installed, you might need to specify NOTHIRD through NOSIXTH to remove existing location information.

Once you specify a contact name using /CONTACT=*name*, you can change the name but you cannot remove it. If you enter /CONTACT=" ", the previously specified contact name remains in effect.

13.6.5.2.2 Verifying Community Information To display the community strings for the OpenVMS host, enter the following command:

```
TCPIP> SHOW CONFIGURATION SNMP /FULL
```

Also, check the community configuration in the TCPIP$VMS_SNMP_CONF.DAT file, as described in Table 13–4.

Make sure that the community string used in the messages matches a valid community of the appropriate type on the server. Check also that the MIB variable is defined with write access and implemented as such in the subagent. Note that in OpenVMS standard MIBS, the Set command is not implemented for some variables defined as writable in the MIB II and Host Resources MIB.

For example, the community must be configured as /TYPE=(READ,WRITE) to process set requests.

If SNMP is not responding to set commands or to other requests:

- One of conditions listed above has not been met.
- The commmunity name is invalid. Check to make sure uppercase or lowercase letters are specified correctly. Community names are case sensitive.
- SNMP is not running on the system.
- There are network, delay, or timeout problems.
- The community address was specified incorrectly.
- Communities with write access are not defined on the server.
- The “public” community configuration was not specified as /TYPE=READ with address 0.0.0.0.
- The SNMP configuration is correct, but SNMP was not restarted after changes were made.

13.6.5.3 Enabling SNMP Version 1 Traps

By default, SNMP sends Version 2 traps, which can be configured using either the TCPIP$CONFIG.COM procedure or the SET CONFIGURATION SNMP command. You can modify SNMP to send Version 1 traps by default, using the trap option described in Table 13–4.

You can implement individual SNMP Version 1 traps even if Version 2 traps are set by default. Add a line for each trap destination to the TCPIP$VMS_SNMP_CONF.DAT file using the following format of the trap option:

```
trap v1 community IP-address[:port]
```

When SNMP Version 1 traps are set by default, you can send SNMP Version 2 traps by adding a line to the TCPIP$VMS_SNMP_CONF.DAT file for each Version 2 trap destination using the following format of the trap option:

```
trap v2c community IP-address[:port]
```

In these formats:

- *community* specifies the community name.
- *IP-address* specifies the IP address of host that is listening for traps.
- *port* specifies the port number. The default port number is 162.

Regardless of the default trap type, you can control the trap type for each trap destination using the appropriate tag (v1 or v2c). For example, the following entries in the TCPIP$VMS_SNMP_CONF.DAT file will cause a Version 1 trap to go to the host with the IP address 120.2.1.2 (community name v1type), and a Version 2 trap to go to the host with the IP address 120.2.2.2 (community name v2type). Both traps will go to the well-known port 162:

```
trap v1 v1type 120.1.2.1
trap v2c v2type 12.2.2.2
```

13.6.6 Solving Management Client Response Problems

When an SNMP client is not getting a response to set, get, getnext, or getbulk requests, even though the SNMP server is configured and running, the problem might be with the operation of the subagent or in the transmission of the query or response message. To test, follow these guidelines:

1. Confirm that TCP/IP Services is running on your host. Enter:

   ```
   TCPIP> SHOW INTERFACE
   ```

   - If TCP/IP Services is not running, a response similar to the following is displayed:

     ```
     %TCPIP-E-INTEERROR, error processing interface request
     -TCPIP-E-NOTSTARTED, TCP/IP Services is not running
     ```

   - If TCP/IP Services is running, a response similar to the following is displayed:

```
                                                    Packets
Interface   IP_Addr          Network mask          Receive          Send     MTU

 WE0        126.65.100.68    255.255.0.0             20298             5    1500
 WF0        126.65.100.108   255.255.0.0             20290             2    4470
 LO0        127.0.0.1        255.0.0.0                3290          3290       0
```

2. To ensure the successful startup of the SNMP master agent and subagents and the operation of the TCPIP$SNMP_REQUEST utility (MIB browser), confirm that the BIND resolver has been configured correctly by entering the following command:

   ```
   TCPIP> SHOW NAME_SERVICE
   ```

   Refer to Chapter 5 for information about configuring the BIND resolver.

3. Check the status of the SNMP service using the following DCL command:

   ```
   SHOW LOGICAL/TABLE=TCPIP$STARTUP_TABLE.
   ```

   This command shows when each TCP/IP Services service startup completed and which user performed each startup. If the SNMP service is not listed, it was either shut down or it was not started.

4. Use the MIB browser on the host to retrieve the OID in question, as described in *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference*.

5. If the local query is successful, use a MIB browser from another host. This is useful when timeout problems indicate that network delays are the cause of the problem.

6. Check the log files for any problems associated with SNMP startup. For detailed information, start the SNMP components separately with tracing enabled, as described in Section 13.6.4.
7. Use a protocol analyzer to intercept messages going to the target. The TCPTRACE utility is available on OpenVMS hosts. Enter the DCL command HELP TCPTRACE for information about how to use this utility. For the failing message:
   - Confirm the community configuration, as described in Section 13.6.5.2.2. Make sure the default community is configured correctly. For example, make sure that a read-only community name, such as "public," is not being used for `set` requests. For more information about community names, refer to the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.
   - Check to make sure the client used the correct query format.
8. Check for problems with ownership, protections, or installation of images, using standard OpenVMS DCL commands, such as DIRECTORY and INSTALL.

   For example, the following message indicates that one of these factors is a possible problem:

   ```
   WARNING: select returned -1 on snmpd sockets: not owner
   ```

   The owner for all SNMP executables should be [SYSTEM]. At a minimum, the protection should be set to S:RWED,O:RWED,G:RE,W:RE.
9. If you cannot get a response for MIB variables handled by certain subagents, verify that the subagents are running by entering the following command:

   ```
   $ SHOW SYSTEM
   ```

   Check for the following processes:
   - TCPIP$SNMP_*n* (master agent)
   - TCPIP$OS_MIBS (standard subagent)
   - TCPIP$HR_MIB (standard subagent)

   See the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide for descriptions of these processes.

   Also check for custom subagents whose process names appear after RUN commands in the following command procedure: SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$EXTENSION_MIB_RUN.COM.

   If these processes and additional subagents follow the model of the Chess example, they should be in LEF state. Excessive time in HIB state indicates a problem. If the processes are not there, check log files for the possible cause of abnormal termination. Note that you must run the SYS$STARTUP:TCPIP$SNMP_SHUTDOWN.COM procedure in order to see entries in the latest .LOG and .ERR files. If a query on members of the `hrFSTable` group results in no response or in a "no such name" response, the problem might be one of the following:
   - No devices are mounted through NFS.
   - Access to mount information is not available because the proxy is not set up to allow the TCPIP$HR_MIB subagent to access NFS-mounted disks.

Additional problems occur if file protections or installation privileges were changed on SYS$SYSTEM:TCPIP$HR_MIB.EXE.

13.6.6.1 Solving Timeout Problems with SNMP Subagents

If queries from a client to an OpenVMS SNMP server are consistently timing out, consider solutions on either the client or server side. For information about checking the client side, refer to the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

On the server:

- Adjust the default timeout value for master agent/subagent communication by redefining the system logical TCPIP$ESNMP_DEFAULT_TIMEOUT, as described in Table 13–5.
- Analyze the performance of slow areas of subagent code to improve the speed of those areas.
- Split up one subagent into multiple subagents, each handling a subset of the original OID tree.
- Adjust the timeout for individual subagents using `esnmp_init( )`, as described in the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

Before making extensive modifications to either the client or the server, consider analyzing the network load for congestion problems.

13.6.7 Disabling SNMP OPCOM Messages

To disable OPCOM messages for SNMP, enter the following command sequence:

```
TCPIP> SET SERVICE SNMP /LOG=NOALL
TCPIP> DISABLE SERVICE SNMP
TCPIP> ENABLE SERVICE SNMP
```

Be aware that when you disable OPCOM messages, you may be suppressing information that is useful for solving problems.

Part 4

Configuring Network Applications

Part 4 describes how to set up popular networking end-user applications and includes the following chapters:

- Chapter 14, Configuring and Managing TELNET, describes how to set your host as a TELNET server, allowing users on remote hosts to establish login sessions.
- Chapter 15, Configuring and Managing FTP, describes how to set up your host as a FTP server, allowing users on remote hosts to transfer files.
- Chapter 16, Remote (R) Commands, describes how to set up the server implementations of the popular Berkeley Remote (R) commands that enable remote file copying (RCP), remote logins (RLOGIN), remote command execution (RSH and REXEC), and remote management of magnetic tape and CD-ROM (RMT/RCD) drives.
- Chapter 17, Configuring and Managing SMTP, and Chapter 18, Configuring and Managing the POP Server, describe how to configure and manage the components that allow users to send and receive internet electronic mail.
- Chapter 19, Configuring XDMCP-Compatible X Displays, describes how to configure an XDMCP-compatible X display using the TCP/IP Services XDM server.

14

Configuring and Managing TELNET

The TCP/IP Services product includes and implementation of the TELNET end-user application.

This chapter describes how to set up your host as a TELNET server.

For information about using TELNET, see the *DIGITAL TCP/IP Services for OpenVMS User's Guide* guide. For information about using the TELNET print symbiont, see Chapter 23.

This chapter describes:

- How to manage the TELNET service (Section 14.1)
- How to solve TELNET problems (Section 14.2)

14.1 Managing TELNET

Managing TELNET includes the following tasks:

- Setting up user accounts
- Creating and deleting sessions
- Displaying login messages

14.1.1 TELNET Startup and Shutdown

The TELNET service can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$TELNET_STARTUP.COM allows you to start up TELNET independently.
- SYS$STARTUP:TCPIP$TELNET_SHUTDOWN.COM allows you to shut down TELNET independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$TELNET_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when TELNET is started.
- SYS$STARTUP:TCPIP$TELNET_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when TELNET is shut down.

14.1.2 Managing TELNET with Logical Names

Table 14–1 lists the logical names you can use in managing the TELNET service.

Table 14–1 TELNET Logical Names

Logical Name	Description
TCPIP$TELNET_VTA	Enables TELNET virtual terminals.

14.1.3 Setting Up User Accounts

Hosts typically run a TELNET server with TELNET client software. Users on client hosts need valid accounts on server hosts before using TELNET to establish a remote session.

If your local host is to be a TELNET server, create OpenVMS accounts for remote users. You can create several individual accounts or one account that many remote users will share.

14.1.4 Creating and Deleting Sessions

You can create and delete TELNET sessions from within a command procedure or interactively. Enter the DCL command TELNET with the /CREATE_SESSION or /DELETE_SESSION qualifier. These qualifiers have the same function as the following commands:

```
TELNET> CREATE_SESSION host port dev-unit
```

```
TELNET> DELETE_SESSION dev-unit
```

For example:

```
$ TELNET /CREATE_SESSION TS405 2002 902
```

You can create a TELNET device that times out after a specified idle period then reconnects when data is written to it. Use the /TIMEOUT qualifier to specify the idle time and the reconnection interval, as described in the following table:

Qualifier	Description
/TIMEOUT	Creates a TELNET device that has the following connection attributes: • NOIDLE—The connection is broken when the device is finally deassigned. The device will automatically reconnect when data is written to it. • IDLE—Specifies the idle time for the device (in the format *hh:mm:ss*). Note that the time has a granularity of 1 second. If the device is idle for at least the specified amount of time, then the connection will be broken. "Idle" means that the device has neither received nor sent any data for the idle period. • NORECONNECTION—The device does not automatically retry reconnections if they fail. • RECONNECTION—When data is written to the device and it is not connected, this value determines the interval between reconnection attempts. For example, if an application writes to a TN with a RECONNECTION value of 0:1:00 and the first connection attempt fails, subsequent connection attempts will be made in 1-minute intervals.
/NOTIMEOUT	Creates a TELNET device that breaks the connection when the device is finally deassigned (the last channel assignment is deassigned).

14.1.5 Displaying Login Messages

To display login and logout messages at the operator's console and log file, enter:

```
TCPIP> SET SERVICE TELNET /LOG=(LOGIN,LOGOUT)
```

14.1.6 TELNET Client (TN3270)

IBM 3270 Information Display System (IDS) terminal emulation (TN3270) lets users make connections to hosts that use IBM 3270 model terminals.

TN3270 has default IBM 3270 IDS function assignments for DIGITAL keyboards. In addition, users can make their own assignments and might ask you for help. TCP/IP Services provides EBCDIC-to-DMCS and DMCS-to-EBCDIC translation tables you can customize. Appendix B describes how to customize and rebuild these translation tables.

For more information about using TN3270, enter the following DCL command:

```
$ HELP TN3270
```

14.2 Solving TELNET Problems

To improve TELNET performance, try modifying some of the internet parameters. These changes might also decrease the use of system resources.

14.2.1 TELNET Characteristics That Affect Performance

The settings for the TELNET systemwide characteristics might affect TCP/IP Services and TELNET performance. To display the TELNET systemwide characteristics, enter:

```
TCPIP> SHOW SERVICE TELNET /FULL
```

The command generates a display similar to the following:

```
Service: TELNET
  State:  Enabled
 Port: 23  Protocol:  TCP  Address:  0.0.0.0
 Inactivity:  1  User_name: Process:  not defined
 Limit:30  Active: 1  Peak:  4
 File: not defined
 Flags:  Listen Priv Rtty
 Socket Opts:  Keepalive
 Receive: 3000  Send:  3000

 Log Opts:  Actv Dactv Conn Error Logi Logo Mdfy Rjct Addr

 File:  not defined

 Security
 Reject msg:  not defined
 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0
```

14.2.2 Requests That Cannot Be Satisfied

The TELNET server sends the following error message for a TELNET login request that cannot be satisfied:

```
SS$_EXQUOTA
```

This error is due to insufficient local resources, such as:

- Too many sessions

  To determine whether this is the cause of the problem, check to see whether the maximum number of concurrent sessions has been exceeded. Enter the following TCP/IP management command:

  ```
  TCPIP> SHOW SERVICE TELNET
  ```

  If the maximum number of concurrent sessions has been exceeded, the display shows:

  PEAK=*limit*

  To increase the number of allowed sessions, enter the following command:

  ```
  TCPIP> SET SERVICE TELNET /LIMIT=n
  ```

- Insufficient OpenVMS nonpaged pool

  To determine whether this is the cause of the problem, check to see whether the OpenVMS nonpaged pool is insufficient for servicing a new TELNET connection. If so, monitor the server.

  To improve any of the parameters, redefine the logical names.

- Excessive OpenVMS login sessions

  To determine whether this is the cause of the problem, check to see whether the limit for maximum OpenVMS sessions has been exceeded. If the current value is not appropriate, redefine it.

Verify that the CHANNELCNT parameter (in SYSGEN) is larger than the number of simultaneous TELNET and RLOGIN sessions that you plan to support.

15
Configuring and Managing FTP

The File Transfer Protocol (FTP) software transfers files between "nontrusted" hosts. Nontrusted hosts require user name and password information for remote logins.

The TCP/IP Services product includes an implementation of the FTP end-user applications.

This chapter describes:

- How to manage the FTP service (Section 15.1)
- How to solve FTP problems (Section 15.2)

For information on using FTP, see the *DIGITAL TCP/IP Services for OpenVMS User's Guide*.

15.1 Managing FTP

Managing FTP consists of the the following tasks:

- Enabling and disabling FTP
- Configuring anonymous FTP
- Defining FTP logical names
- Monitoring FTP with FTP log files

15.1.1 Enabling and Disabling FTP

After FTP is configured by TCPIP$CONFIG, the postinstallation configuration procedure, it is started automatically when TCP/IP Services is started.

To stop any new connections without losing existing connections, disable the FTP server interactively using the SET NOSERVICE command. To disable FTP when TCP/IP Services starts, use the SET CONFIGURATION NOSERVICE command.

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* for descriptions of the SET SERVICE and SET CONFIGURATION SERVICE commands.

15.1.2 Configuring Anonymous FTP

Anonymous FTP is an FTP session in which a user logs in to the remote server using the user name ANONYMOUS and, by convention, the user's real user name as the password.

On the local FTP server, local users can access files without password authentication. Remote users do not require an account. File access is controlled by regular OpenVMS access restrictions.

When you use TCPIP$CONFIG to establish an ANONYMOUS account, a new account is created with the UIC [TCPIP,ANONYMOUS] (numeric [3375,*xx*]), user name ANONYMOUS, account ANONY, default directory SYS$SYSDEVICE:[ANONYMOUS], and the following types of login access:

network	full access
batch	no access
local	no access
dialup	no access
local	no access

The usual OpenVMS file protection codes restrict file access for inbound anonymous FTP sessions to this directory, its subdirectories, and files with an owner attribute of [TCPIP,ANONYMOUS].

When the ANONYMOUS account has been created, a remote FTP client can:

- Copy files to and from GUEST$PUBLIC.
- From the ANONYMOUS$USER directory:
  - Delete files
  - Create directories
  - Delete directories
  - Rename files
  - Rename directories

You can set up guest and public directories for bulletin board or group interest. Make sure the directory protections are set to read-only or read/write, as needed.

In the following example, UNIX user `ubird` connects to the ANONYMOUS account on OpenVMS host TRAGOPAN. TRAGOPAN asks for `ubird`'s password, which is not echoed. In response to this request, the user should supply the local system user name for identification purposes.

```
% ftp tragopan

Connected to tragopan.asian.pheasant.edu.
220 tragopan.asian.pheasant.edu FTP Server (Version 5.1) Ready.

Name (tragopan:wings): ANONYMOUS

331 Guest login ok, send ident as password.
Password: CARIBBEAN

230  Guest login ok, access restrictions apply.

        Welcome to Compaq TCP/IP Services for OpenVMS
        on internet host TRAGOPAN    Date 24-JUN-2000
FTP>
```

15.1.2.1 Concealed File Systems

The FTP server processes each command individually as it receives the command and displays a reply based on the command parameters. A reply can include a file specification that displays part of the server file system.

For security, anonymous FTP masks file system devices and directories in FTP replies. The following messages show the difference between an unmasked file structure, shown in the first reply, and the less-specific, masked structure in the second reply.

```
220 opening data connection for USER8$:[HIDEME.PROJECT.TASK]PLAN.PS
220 opening data connection for SYS$LOGIN:[PROJECT.TASK]PLAN.PS
```

15.1.2.2 Setting Up Anonymous FTP

Complete the following steps to set up anonymous FTP access on your system:

1. Use the TCPIP$CONFIG procedure to create an account named ANONYMOUS with the password GUEST.

   To create the ANONYMOUS user account, select Optional Components from the main menu, then select Setup Anonymous FTP Account and Directories.

2. Set account access restrictions NOLOCAL, NOBATCH, NOREMOTE, and NODIALUP.

3. Create a welcome banner.

   When a user logs in, FTP displays a welcome that you can specify by entering the text into a text file and defining the TCPIP$FTP_SERVER_ANNOUNCE logical to point to the text file. Make sure the protection on the file is world readable (W:R). For example:

   ```
   $ DEFINE/SYSTEM/EXEC TCPIP$FTP_SERVER_ANNOUNCE -
   _$ "@SYS$SYSDEVICE:[TCPIP$FTP]ANNOUNCE.TXT"
   ```

   When an anonymous user logs in, FTP informs the user of the account's restrictions. Enter the text you want to display into a text file. Then define the TCPIP$FTP_ANONYMOUS_WELCOME logical name to point to the text file. Make sure the text file protection is world readable (W:R). For example:

   ```
   $ DEFINE/SYSTEM/EXEC TCPIP$FTP_ANONYMOUS_WELCOME -
   _$ "@SYS$SYSDEVICE:[TCPIP$FTP]WELCOME.TXT"
   ```

4. Specify a location for the log files generated by FTP sessions.

   Use the TCPIP$FTP_ANONYMOUS_LOG logical name. If you do not define TCPIP$FTP_ANONYMOUS_LOG, FTP puts the files in SYS$SYSDEVICE:[TCPIP$FTP]TCPIP$FTP_ANONYMOUS.LOG.

5. Specify a user name for the anonymous FTP account. Define the logical name TCPIP$FTP_ANONYMOUS_ALIAS. See Table 15–1 for more information.

15.1.3 Managing FTP with Logical Names

Table 15–1 lists the logical names that you can use to manage the FTP server.

Table 15–1 FTP Logical Names

Logical Name	Description

TCPIP$FTP_ANONYMOUS_ALIAS	Defines an equivalence list (up to 10 entries) of the login names of users with access to the ANONYMOUS account. These users share the same access rights and restrictions.
	If you do not define this logical name, the default is ANONYMOUS as the only login name.
	The following command shows how to create an equivalence list with the names THOMAS, JONES, and SMITH. These users can log in to the ANONYMOUS account without a password.
	`$ DEFINE/SYSTEM/EXEC TCPIP$FTP_ANONYMOUS_ALIAS -`
	`_$ THOMAS,JONES,SMITH`

TCPIP$FTP_SERVER_ANNOUNCE	Defines location and file name for the announcement text displayed to users when they connect, before the login sequence. The following example shows a prelogin announcement: `"Compaq TCP/IP Services for OpenVMS  FTP Server Ready."`
TCPIP$FTP_ANONYMOUS_WELCOME	Defines location and file name for the welcome text displayed to anonymous users at connect time, after the login sequence.
TCPIP$FTP_CONVERT_FILE	Define this logical name as TRUE or FALSE. If defined as TRUE, the FTP server converts files to variable with fixed-length control (VFC) formatted files before transfer. With the VFC file, users retain the Record Management Services (RMS) formatting information of their files. For more information about RMS, refer to the *OpenVMS Record Management Services Reference Manual*. If TCPIP$FTP_CONVERT_FILE is defined as FALSE, there is no conversion, and RMS formatting information is lost after the file transfer.
TCPIP$FTP_ANONYMOUS_ DIRECTORY	Displays public directories accessible by the anonymous FTP user.
TCPIP$FTP_FILE_ALQ	Specifies the number of blocks to be preallocated by Record Management Services (RMS) to a disk when a file is created.
TCPIP$FTP_FILE_DEQ	Specifies the number of blocks to be added when RMS automatically extends the file.
TCPIP$FTP_ANONYMOUS_LOG	Defines the location of the anonymous log file. The default is SYS$SYSDEVICE:[TCPIP$FTP].
TCPIP$FTPD_IDLETIMEOUT	Defines the maximum time interval that FTP child processes can remain idle before FTP closes them. TCP/IP Services terminates the FTP process if no control or data connection activity exists for the specified time. The default idle time is 15 minutes. This feature can help to improve system performance. Specify the value as *hh:mm:ss*.
TCPIP$FTP_KEEPALIVE	Detects idle and broken FTP connections. Define it on the client host as TRUE or FALSE.
TCPIP$FTPD_KEEPALIVE	Detects idle and broken FTP connections. Define it on the server host by entering: `TCPIP> SET SERVICE FTP /SOCKET_OPTIONS=KEEPALIVE`

(continued on next page)

Table 15–1 (Cont.) FTP Logical Names

Logical Name	Description
TCPIP$FTP_NO_VERSION	If defined, FTP does not send file version numbers when you enter the `mget` and the `ls` commands to a host that is not an OpenVMS host. Define this logical name in the system logical name table, as follows: `$ DEFINE /SYSTEM/EXEC TCPIP$FTP_NO_VERSION 1`
TCPIP$FTP_RAW_BINARY	With this logical name turned on, FTP transfers files in block I/O mode if the server and client are in binary (image) mode. To activate this feature, define the logical name as TRUE. An FTP end-user can override your FALSE definition with the FTP PUT /RAW command.
TCPIP$FTP_STREAMLF	If defined, the FTP server and client create files as RMS STREAM_LF files. The default is variable-length files.
TCPIP$FTP_WNDSIZ	Sets the size of the TCP send and receive transmission windows. Specify a decimal number for the number of bytes.
TCPIP$FTP_SERVER_LOG_CLIENT_BY_ADDRESS	Specifies that the FTP server will be using IP addresses instead of host names.
TCPIP$FTP_SERVER_NAME_SERVICE_TIMEOUT	Specifies the number of seconds for the timeout interval. For more information, refer to the description of the SET NAME_SERVICE/TIMEOUT command in the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual. This logical name has no effect if the FTP server is using IP addresses instead of host names (that is, the logical name TCPIP$FTP_SERVER_LOG_CLIENT_BY_ADDRESS is defined).
TCPIP$FTP_SERVER_NAME_SERVICE_RETRY	Specifies the number of times the BIND resolver should attempt to contact a BIND server if the first attempt fails. This logical name has no effect if the FTP server is using IP addresses instead of host names (that is, the logical name TCPIP$FTP_SERVER_LOG_CLIENT_BY_ADDRESS is defined).

15.1.3.1 FTP Log Files

By default, the FTP server creates several log files you can use to monitor the service and user transactions. These log files are:

- SYS$SYSDEVICE:[TCPIP$FTP]TCPIP$FTP_RUN.LOG

  This log contains an abbreviated dialog of each new connection process. It is created by each new invocation of the server and is accessible only after an ongoing connection times out or after being closed by the user.

- SYS$SYSDEVICE:[TCPIP$FTP]TCPIP$FTP_ANONYMOUS.LOG

  This log contains Anonymous FTP entries that show:

  - The user name and source host (FTP client) for the session
  - The time the session was initiated and terminated
  - The FTP command that was entered
  - A time notation for the command
  - The source and destination file names

- SYS$LOGIN:FTP_SERVER_RUN.LOG

  This log is created in the user's default login directory.

The number of log files (one per FTP session) might become large. To limit the number of versions, enter:

```
$ SET FILE file /VERSION=n
```

15.1.3.2 FTP Startup and Shutdown

The FTP service can be shut down and started independently from TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$FTP_STARTUP.COM allows you to start up FTP independently.
- SYS$STARTUP:TCPIP$FTP_SHUTDOWN.COM allows you to shut down FTP independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$FTP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when FTP is started.
- SYS$STARTUP:TCPIP$FTP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when FTP is shut down.

15.2 Solving FTP Problems

You can improve FTP performance for users who transfer large files from systems that are not running TCP/IP Services to a host running the TCP/IP Services software.

15.2.1 Performance

Large file transfers can affect file transfer performance. A file transfer consists of the following events:

1. FTP calls RMS to create the file.
2. RMS creates the file with the system's default for number of blocks to be allocated (FTP_FILE_ALQ value).
3. If the file being copied is larger than the space originally allocated, RMS extends the space by adding blocks of memory space.
4. The number of extension blocks is determined by the system's RMS default extension quantity (FTP_FILE_DEQ value). For more information about RMS, refer to the *OpenVMS Record Management Services Reference Manual*.

Performance is affected by the RMS overhead taken up by the file extension process. One way to improve performance is to reset the appropriate parameters. To do this, redefine the FTP logical names that:

- Reset buffer sizes
- Preallocate disk blocks

- Increase the inactivity timer

These logical names are described in the following sections.

15.2.1.1 Buffer Sizes

Changing the window size of the send and receive buffers can improve network performance. To set or modify the window size, define or redefine the logical name TCPIP$FTP_WNDSIZ.

- For a systemwide change, redefine this logical name in the system table.

  Edit the SYS$MANAGER:TCPIP$SERVICES_SETUP file to add this line:

  ```
  $ DEFINE /SYSTEM /EXEC TCPIP$FTP_WNDSIZ 4096
  ```

- For the change to apply to one user, define the logical name in the LOGIN.COM file in the default directory of that user.

For noisy lines, such as modems, you should set the value of the TCPIP$FTP_ WNDSIZ parameter to a lower number.

15.2.1.2 File Allocation and Extension Sizes

FTP logical names preallocate disk blocks. FTP tells RMS to truncate unused blocks so that disk space is not wasted. This can affect RMS performance.

To reduce the RMS overhead, use the following logical names:

- TCPIP$FTP_FILE_ALQ — Modifies the allocation quantity.

  Specifies the number of blocks to be allocated to a disk file when it is created. For example:

  ```
  $ DEFINE /SYSTEM/EXEC TCPIP$FTP_FILE_ALQ 50000
  ```

- TCPIP$FTP_FILE_DEQ — Default extension quantity.

  Specifies the number of blocks to be added when RMS automatically extends the file. For example,

  ```
  $ DEFINE TCPIP$FTP_FILE_DEQ 100
  ```

Define these logicals in the TCPIP$SYSTARTUP.COM procedure, or in the SYS$MANAGER:STARTUP_VMS.COM file before the command that starts TCP/IP Services. Because disk quotas may control the system, these logical names are defined by default as zero (system RMS defaults) or are undefined. For file transfers between hosts that both use VMS Plus mode, these logical names have no effect.

15.2.1.3 Inactivity Timer

The larger the inactivity timer value, the longer FTP maintains sessions without timing out. Excessive inactive sessions might slow down performance, degrade security, or prevent other users from establishing sessions.

To increase the inactivity timer, change the value of the TCPIP$FTPD_IDLETIMEOUT logical name. The default is 15 minutes. For example:

```
$ DEFINE TCPIP$FTPD_IDLETIMEOUT 01:00:00
```

16
Remote (R) Commands

The TCP/IP Services software includes client and server implementations of the Berkeley Remote (R) command applications: RCP, RLOGIN, RSH, REXEC, and RMT/RCD. These applications provide end users with the following capabilities:

RCP	Allows files to be copied between remote hosts.
RLOGIN	Provides interactive access to remote hosts.
RSH	Passes a command to a remote host for execution.
REXEC	Authenticates and executes RCP and other commands.
RMT/RCD	Provides remote access to magnetic tape and CD-ROM drives.

This chapter reviews key concepts and describes:

- How to manage the R command servers (Section 16.2)
- Security considerations (Section 16.3)
- How to create a welcome message (Section 16.4)
- How the Remote Magnetic Tape/Remote CD-ROM (RMT/RCD) service operates (Section 16.5)

For information about using these applications, see the *DIGITAL TCP/IP Services for OpenVMS User's Guide*.

16.1 Key Concepts

In addition to password authentication, the R commands use a system based on **trusted hosts** and users. Trusted users on trusted hosts are allowed to access the local system without providing a password. Trusted hosts are also called "equivalent hosts" because the software assumes that users given access to a remote host should be given equivalent access to the local host. The system assumes that user accounts with the same name on both hosts are "owned" by the same user. For example, the user logged in as `molly` on a trusted system is granted the same access as a user logged in as `molly` on the local system.

This authentication system requires databases that define the trusted hosts and the trusted users. On UNIX systems, these databases are:

- `/etc/hosts.equiv`

  This file defines the trusted hosts and users for the entire system.

- `.rhosts`

  This file defines the trusted hosts and users for an individual user account. This file is located in the user's home directory.

On OpenVMS hosts, the proxy database TCPIP$PROXY.DAT defines trusted hosts and users for the entire system.

16.2 Managing the R Command Servers

The following sections describe the command procedures and logical names used in managing the R command servers.

16.2.1 R Command Server Startup and Shutdown

Each R command server can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted.

The following files allow you to start up each R command server independently:

- SYS$STARTUP:TCPIP$REXEC_STARTUP.COM
- SYS$STARTUP:TCPIP$RMT_STARTUP.COM
- SYS$STARTUP:TCPIP$RSH_STARTUP.COM
- SYS$STARTUP:TCPIP$RLOGIN_STARTUP.COM

The following files allow you to shut down the each R command server independently:

- SYS$STARTUP:TCPIP$REXEC_SHUTDOWN.COM
- SYS$STARTUP:TCPIP$RMT_SHUTDOWN.COM
- SYS$STARTUP:TCPIP$RSH_SHUTDOWN.COM
- SYS$STARTUP:TCPIP$RLOGIN_SHUTDOWN.COM

To preserve site-specific parameter settings and commands to be executed when the R server starts up, create one of the following files, as appropriate. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$REXEC_SYSTARTUP.COM
- SYS$STARTUP:TCPIP$RMT_SYSTARTUP.COM
- SYS$STARTUP:TCPIP$RSH_SYSTARTUP.COM
- SYS$STARTUP:TCPIP$RLOGIN_SYSTARTUP.COM

To preserve site-specific parameter settings and commands to be executed when the R server shuts down, create one of the following files, as appropriate. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$REXEC_SYSHUTDOWN.COM
- SYS$STARTUP:TCPIP$RMT_SYSHUTDOWN.COM
- SYS$STARTUP:TCPIP$RSH_SYSHUTDOWN.COM
- SYS$STARTUP:TCPIP$RLOGIN_SYSHUTDOWN.COM

16.2.2 Managing RLOGIN with Logical Names

Table 16–1 lists the logical names you can use for managing the RLOGIN service.

Table 16–1 RLOGIN Logical Names

Logical Name	Description
TCPIP$RLOGIN_VTA	Enables RLOGIN virtual terminals. For more information, see Section 16.3.
TCPIP$RLOGIN_MESSAGE	Specifies the welcome message displayed by the RLOGIN server. For more information, see Section 16.4.

16.3 Security Considerations

Because R commands can bypass normal password verification, it is important to configure these applications carefully to avoid compromising system security. In a complex networking environment, improperly configured R commands can open access to your host to virtually anyone on the network.

A properly configured environment grants remote access to preauthorized clients. You can limit access by adding an entry to the proxy database (TCPIP$PROXY.DAT) for each user authorized to access your host. This entry, called a communication proxy, provides the user name and name of the remote host. To add a communication proxy, enter:

```
TCPIP> ADD PROXY user /HOST=host /REMOTE_USER=user
```

For each host, be sure to define the host name and any aliases.

Users with communication proxies cannot use virtual terminals. Therefore, if the logical name TCPIP$RLOGIN_VTA is set, users logging in by proxies will observe that the terminal device they are assigned is displayed as TNA*nnn* rather than VTA*nnn*. For more information, see Section 16.2.2.

16.3.1 Registering Remote Users

For users on UNIX hosts, the following information must be listed in at least one of the following databases:

Database File	Type of Information
`/etc/hosts.equiv`	Host name and user name
`.rhosts` (in the user's home directory)	Host name and user name

For users on OpenVMS clients running TCP/IP Services, check that the appropriate proxy information is in the remote system's proxy database.

You can also restrict remote printing to specific users by entering:

```
TCPIP> SET SERVICE service /FLAGS=APPLICATION_PROXY
```

With this flag set, the R commands use the communication entries in the proxy database for authentication.

To reject access from a remote host, use the SET SERVICE *service* /REJECT command. For example:

```
TCPIP> SET SERVICE RLOGIN /REJECT=HOSTS=(loon,ibis,tern)
```

16.3.2 Case-Sensitivity Flag

The proxy database is case sensitive for remote user names. The case you use for communications entries affects the way users access your host, so use case in a consistent fashion. In the proxy database, if the user name is in:

- Uppercase, the user must use the /NOLOWERCASE qualifier.
- Lowercase, RSH and RLOGIN default to /LOWERCASE.

If the flag CASE_INSENSITIVE is set, the server matches an incoming user name with an all-lowercase or an all-uppercase remote user name in the proxy database.

The case-sensitivity flag for RLOGIN, RSH, and RCP defaults to CASE_ INSENSITIVE. With this setting, the server accepts both all-uppercase and all-lowercase user names.

Ensure that RSH is enabled, because no RCP service exists. Instead, RCP uses the RSH server process. (RCP uses RSH or REXEC to do its work. RSH must be configured properly for RCP to work.)

16.4 Creating a Welcome Message

To modify the welcome message displayed by the RLOGIN server, define the TCPIP$RLOGIN_MESSAGE logical name and specify the text. For example, the following command defines a welcome message for RLOGIN clients when they log in to the server:

```
$ DEFINE /SYSTEM TCPIP$RLOGIN_MESSAGE "OpenVMS RLOGIN Server Version 5.1"
```

16.5 Remote Magnetic Tape and Remote CD-ROM (RMT/RCD)

The Remote Magnetic Tape/Remote CD-ROM (RMT/RCD) server provides remote system access to local OpenVMS magnetic tape and CD-ROM drives. The tape or CD-ROM drives appear to the RMT client users as if they were mounted locally. The RMT server fully implements the UNIX commands `rdump` and `rrestore` and the OpenVMS commands MOUNT, BACKUP, COPY, and EXCHANGE.

This section assumes that you are familiar with device mounting and server access conditions relevant to the R command services.

16.5.1 Preparing Drives for Remote Mounts

Perform the following tasks to make sure the remote client can access the tape or CD-ROM drive:

1. Enable the RSH, REXEC, and RMT services.
2. Load a magnetic tape or CD-ROM into the device.

   With a tape device, the client mounts and allocates the tape; you do not need to perform this task.

   With a CD-ROM device, you need to make the device accessible by entering a MOUNT /SYSTEM command.
3. Make sure the remote shell command (RSH) works from the UNIX root account.
   - Create the OpenVMS account named ROOT. This account must have PHYIO and VOLPRO privileges.

- Create a communication proxy that associates the remote RMT client user with the OpenVMS account ROOT on the RMT server host. For example:

  ```
  TCPIP> add proxy root /HOST=host /REMOTE=user
  ```

  See Section 16.3 for more information about communication proxies.

4. Make sure the rsh command works from the user's account on the remote UNIX host.
5. For the OpenVMS account ROOT, suppress SYS$LOGIN and LOGIN.COM output by entering the following commands:

   ```
   $ RMT_VERIFY = 'F$VERIFY(0)
    . . .
   $ IF (F$MODE() .NES. "OTHER") THEN $RMT_VERIFY = F$VERIFY(RMT_VERIFY)
   ```

16.5.2 Client Utilities

On the remote host, a user can use rdump to dump files to OpenVMS tapes, or rrestore to restore files from OpenVMS tapes. The functionality of rdump and rrestore depends entirely on the type of UNIX system you use and not on the RMT service. For example, not all UNIX systems let you restore files selectively using rrestore.

When you enter these remote dump and restore commands, you must specify either a valid OpenVMS magnetic-tape device name, or a file name.

See the sections on dump, rdump, restore, and rrestore in your client system's documentation for details. Be careful about the order in which you specify options on the command line.

Here is an example of an rdump command:

```
> /etc/rdump 0f lilac:mua0:/nomount/density=1600 /usr
```

In the example, the remote user requests to remotely dump the /usr file system onto device mua0: on system lilac and specifies the /nomount qualifier and a tape density of 1600 bits per inch.

You can specify the qualifiers described in Table 16–2 with magnetic-tape device names.

Table 16–2 RMT Magtape Qualifiers

Qualifier	Description
/[NO]ASSIST	Specifies whether to use operator assistance to mount the volume. The default is /NOASSIST.
/BLOCKSIZE=*n*	Specifies the block size for magnetic tape volumes. The default is 65534 bytes.
/CD	Indicates that the remote device is a CD-ROM device.
/COMMENT=*"string"*	Specifies additional information included with the operator request when the mount operation requires operator assistance (/ASSIST). The comment appears in the OPCOM message for the operator.

(continued on next page)

Table 16–2 (Cont.) RMT Magtape Qualifiers

Qualifier	Description
/DENSITY=*n*	Specifies the density (in bits per inch) at which to write a foreign or unlabeled magnetic tape. The default is the current density.
/[NO]MOUNT	Specifis whether to use the OpenVMS MOUNT service to mount the tape. /NOMOUNT gains access to the tape directly without mounting it. Use this for UNIX utilities that expect the tape drive to hold its position (not rewind) if the utility closes it. The default is /MOUNT.
/[NO]REWIND	Specifies whether to rewind the drive when it is closed. The default is /REWIND.
/[NO]STREAM	Specifies whether to read the tape in record mode (/NOSTREAM) or byte-stream mode (/STREAM). The default is /STREAM.
/[NO]UNLOAD	Specifies whether to unload the drive when it is closed. The default is /UNLOAD.
/[NO]WRITE	Specifies whether you can write to the magnetic tape. The default is /WRITE.

16.5.3 Client Examples

The following steps perform `rdump` and `rrestore` functions from a UNIX client system. These commands dump two UNIX directories to the tape with separate `rdump` commands. These commands then restore files selectively from the tape to the UNIX client system:

1. Put the directories on the tape by entering two `rdump` commands. Specify the `/nomount/norewind/nounload` qualifiers to prevent OpenVMS from rewinding the tape. Although the UNIX system reports that the tape was rewound, it was not actually rewound. The commands are:

   ```
   UNIX> /etc/rdump 0f vax:device/nomount/norewind/nounload dir1
   UNIX> /etc/rdump 0f vax:device/nomount/norewind/nounload dir2
   ```

2. Restore the files selectively from the tape using `rrestore`. Be sure the tape is loaded and rewound. Use either the interactive or noninteractive command syntax.

The `rrestore` command might display messages such as `"You have not read any volumes yet"` and then ask you to specify the next volume. Although these messages might appear, `rrestore` should work properly.

In the following example, `rrestore` extracts the file specified by *file_name* from dump file number 2 on the tape:

```
UNIX> /etc/rrestore fsx vax:device/nomount/nounload/norewind 2 file-name
```

In the following example, `rrestore` invokes the interactive utility to let the user specify particular files that were put on the tape in dump file 2. The `add` command then adds the files to the extraction list and the `extract` command restores them:

```
UNIX> /etc/rrestore fis vax:device/nomount/nounload/norewind 2
restore> add file_name
restore> extract
```

17

Configuring and Managing SMTP

The Simple Mail Transfer Protocol (SMTP) is a standard protocol that provides a reliable and efficient mail delivery system between systems communicating in a TCP/IP network. SMTP specifies the format of control messages sent between two machines to exchange electronic mail, but it does not specify the mail interface.

The TCP/IP Services product implements SMTP as an OpenVMS symbiont that works with the OpenVMS Mail utility.

This chapter reviews key concepts and describes:

- How to configure SMTP (Section 17.2)
- How to create a local alias file (Section 17.3)
- How to manage SMTP (Section 17.4)
- How to modify the SMTP configuration (Section 17.5)
- How to configure the SMTP antiSPAM feature (Section 17.6)
- How to manage the SMTP send-from-file (SFF) feature (Section 17.7)
- How to disable the SMTP outbound alias feature (Section 17.8)
- How to solve SMTP problems (Section 17.9)

See the *DIGITAL TCP/IP Services for OpenVMS User's Guide* for information about using SMTP to send and receive mail.

17.1 Key Concepts

To be reliable, electronic mail systems must be able to cope with situations where the recipient is temporarily unavailable, for example, if the recipient's host is down or off line. Mail must also be able to handle situations where some of the recipients on a distribution list are available and some are not.

SMTP is a store-and-forward mail protocol that accepts mail from an originating host and forwards it through one or more intermediate hosts before it reaches its final destination. Note that this behavior differs from OpenVMS Mail, where mail is sent directly from the originating node to the destination node.

17.1.1 How SMTP Clients and Servers Communicate

In most implementations, SMTP servers listen at port 25 for client requests. In the TCP/IP Services implementation of SMTP, the SMTP receiver is invoked by the auxiliary server when an inbound TCP/IP connect comes in to port 25 (if the SMTP service is enabled). The auxiliary server runs the command procedure specified in the SMTP service database entry that runs the receiver. The receiver image is SYS$SYSTEM:TCPIP$SMTP_RECEIVER.EXE. The receiver process runs in the TCPIP$SMTP account.

The SMTP symbiont processes all mail on the host. It receives jobs one at a time from the generic SMTP queue and delivers them either locally by means of OpenVMS Mail, or remotely by means of SMTP.

The configuration procedure TCPIP$CONFIG sets up the SMTP queues for you. See Section 17.2 for more information on configuring SMTP.

After receiving a client request, the SMTP server responds, indicating its status (available or not available). If the server is available, it starts an exchange of control messages with the client to relay mail. (Like FTP, SMTP does not define a message format. SMTP commands are sent as ASCII text, and the SMTP server at the remote host parses the incoming message to extract the command.)

The following steps occur:

1. The auxiliary server listens for requests, starts the SMTP receiver, and accepts the TCP connection.
2. The client identifies itself by sending its fully qualified domain name.
3. The server replies with its own fully qualified domain name.
4. The client sends the full e-mail address of the sender enclosed in angle brackets; if the server is able to accept the mail, it returns a readiness code.
5. The client sends the full mail address (also enclosed in angle brackets) of the message's intended recipient.
6. The client sends the body of the message.

A minimum of five control message commands are required to conduct the preceding sequence. Table 17–1 describes these commands.

Table 17–1 SMTP Client Commands

Command	Description
HELO	Identifies the originating host to the server host. Use the /DOMAIN qualifier to provide the name of the originating host.
MAIL FROM:<*reverse-path*>	Identifies the address at which undeliverable mail should be returned. Usually is the originating host.
RCPT TO:<*forward-path*>	Address of the intended receiver. If sending mail to multiple recipients, use one RCPT TO command for each recipient.
DATA	Signals the end of the RCPT TO commands and tells the recipient to prepare to receive the message itself.
QUIT	Indicates no more commands.

These commands are described in RFC 821.

17.1.2 Understanding the SMTP Control File

With TCP/IP Services SMTP, each mail message is packaged into a special-purpose binary file called a control file. This control file is submitted to a generic SMTP queue to be processed by the SMTP symbiont. Each control file contains one SMTP mail message. Note that an SMTP message addressed to multiple recipients is stored in one control file.

Control file names provide information about the mail contained within. The format for the control file name is as follows:

yymmddmmshh_user-name.TCPIP_*scnode*

where:

yymmddmmshh is the timestamp taken when the file is created.

user-name is the user name of the process in which the control file was created. Values for this name include:

- TCPIP$SMTP — The mail arrived through SMTP. The file was created by the SMTP receiver process running in the TCPIP$SMTP account.
- MAIL$SERVER — The mail arrived over DECnet and was destined for an SMTP address. In this case, the control file is created by the DECnet MAIL11 network object that runs the MAIL$SERVER account. This happens when the user sets mail forwarding to an SMTP address.
- SYSTEM — If the control file is in the TCPIP$SMTP account directory, this indicates the message is either undeliverable mail, or a mail message from the SYSTEM account.
- *username* — Mail composed by the user and sent to an SMTP address.

scnode is the value of the SYSGEN SCSNODE parameter.

Control files are written to the TCPIP$SMTP account's login directory. The default directory for this account is SYS$SPECIFIC:[TCPIP$SMTP].

17.1.3 Understanding OpenVMS Mail Headers

The OpenVMS Mail utility contains up to four headers on a mail message:

- `From:`
- `To:`
- `Subj:`
- `CC:`

SMTP supports a large set of mail headers, including:

- `Resent-Reply-To:`
- `Resent-From:`
- `Reply-To:`
- `Resent-Sender:`
- `Sender:`
- `ReturnPath:`

When it composes the OpenVMS Mail message, SMTP uses the text from the first SMTP header in the list that it finds for the OpenVMS Mail `From` header.

17.1.4 Understanding SMTP Addresses

SMTP addresses are of the form *userID@domain.name*, where *domain.name* refers to a domain for which there is a DNS MX record. Mail exchange (MX) records tell SMTP where to route the mail for the domain.

17.1.5 How SMTP Routes Mail

To find a destination address, SMTP routing looks up addresses in this order:

1. Local MX database
2. BIND MX records
3. BIND A records
4. Local hosts database

Most messages are routed using the BIND records. Local MX records are useful if you want to customize your system's mail routing. DNS-based records are networkwide. If you have local MX records, remember that they are case sensitive and are available on the local node only.

17.1.5.1 Using Local MX Records

SMTP uses the information stored in local MX records, if available, to route mail. MX tells the SMTP where to route mail for a particular destination domain. The DNS (such as BIND) maintains the MX records, but SMTP makes use of them. Each MX record contains the following fields:

Destination domain	Matches the *domain* portion of the address. This is the `key` field of the MX record. For example, if mail is to be sent to `jones@xyzcorp.com`, MX lookup is done on the destination domain `xyzcorp.com`. Multiple MX records for the same destination are allowed. Therefore, in a sense, the destination domain field allows duplicate keys.
Gateway host name	Specifies the name of the host through which mail sent to the destination domain should be routed.
Preference	Prioritizes multiple MX records for the same destination domain. The lower the preference value, the higher the priority for the MX record. That is, lower-preference MX records are attempted before higher-preference records. Multiple MX records to the same domain can have the same preferences.

Creating multiple MX records for the same destination domain provides the following advantages:

- Enables load balancing between mail routers. In this case, use the same preferences for all the MX records with the same destination domain.
- Ensures that mail can still be delivered even if one of the routers becomes unavailable.
- Provides MX-based routes for mail inside and outside a firewall.

Use the SET MX_RECORDS command to enter routing information to the MX database. For example, the following command assigns MERLIN as the gateway for CROW with a preference 100:

```
TCPIP> SET MX_RECORDS CROW /GATEWAY=MERLIN /PREFERENCE=100
```

MX routing first checks the local preference but tries it only once in the lookup process.

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* for a detailed desciption of the SET MX_RECORDS command.

17.1.5.2 Using SMTP Zones and Alternate Gateways

When configuring SMTP, you supply the name of the domain for your environment with the /ZONE qualifier to the SET CONFIGURATION SMTP command. If you do not supply a domain name, the zone defaults to one level higher than your local domain. For example, if the fully qualified domain name is `a.b.com`, the default value of /ZONE is `b.com` (assuming that, because TCPIP has been started, the domain is known).

Mail for delivery outside of your zone is sent to its destination by the alternate gateway, as defined by the /GATEWAY qualifier. If you define an alternate gateway, SMTP routes mail to destinations outside the SMTP zone defined on the alternate gateway. SMTP uses MX records for routing mail within the zone and, if no alternate gateway is defined, elsewhere as well.

The following example defines the alternative gateway MY.ALT.MYZONE.COM and the zone MYZONE.COM.

```
TCPIP> SET CONFIGURATION SMTP/GATEWAY=ALTERNATE=MY.ALT.MYZONE.COM
TCPIP> SET CONFIGURATION SMTP/ZONE=MYZONE.COM
```

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for a detailed desciption of the SET CONFIGURATION SMTP command.

To send mail to the alternate gateway, SMTP does an MX lookup on the alternate gateway and uses the resulting list of MX records to get the mail to the alternate gateway. To understand the advantages of this method over a simple lookup of A records, consider the following example.

The alternate gateway and zone are configured as follows:

```
TCPIP> SHOW CONFIGURATION SMTP
 . . .
Alternate gateway:  relay.abc.com
 . . .
Zone:               abc.com
 . . .
```

Further, there is no A record configured for the destination domain `relay.abc.com`. Therefore, `relay.abc.com` is not a valid host name. This is demonstrated by the following command:

```
TCPIP> SHOW HOST RELAY.ABC.COM
%TCPIP-W-NORECORD, Information not found
-RMS-E-RNF, record not found
```

There is no such host as `relay.abc.com` because `relay.abc.com` is only an MX destination domain with multiple records at the same preference.

MX records have been set up accordingly. For example:

```
TCPIP> SHOW MX RELAY.ABC.COM

                                BIND MX database

   Server:          1.2.3.4          host.abc.com

   Gate address     Preference       Gate name

   1.3.4.5          100              mail11.abc.com
   1.3.5.6          100              mail13.abc.com
   2.4.5.6          200              mail2.abc.com
   2.4.5.7          200              mail1.abc.com
   3.4.5.6          300              mail21.abc.com
   3.4.6.7          300              mail12.abc.com
```

In this example, when SMTP receives a mail message destined for a domain outside of the abc.com domain, it uses the list of MX records to send the mail to the entity called relay.abc.com. Even when mail is routed through the alternate gateway, the MX lookup list is used.

This type of configuration provides redundancy. Even if one or more of the systems pointed to by the MX records is down, mail can be routed through one of the systems that is running.

If the alternate gateway was reached through a simple A record (hostname) lookup and the host was down or could not be reached, all outbound mail outside the zone would have to wait until the host came back on line.

You can define the alternate gateway using an IP address; this bypasses the MX lookup logic. For example:

```
CPIP> SET CONFIGURATION SMTP/ALTERNATE=GATEWAY=1.2.3.4
```

In this case, all mail destined for the alternate gateway will go to the specified IP address (1.2.3.4) with no MX lookup.

17.2 Configuring SMTP

Use the configuration procedure TCPIP$CONFIG to set up SMTP on your host. If you need to reconfigure or further refine your SMTP environment, use the SET CONFIGURATION SMTP command. With this command, you can change the way SMTP:

- Relays messages
- Determines the route
- Determines how many times it retries a relay and the length of time between delivery attempts
- Sends and receives timeouts

For a complete description of this command, its qualifiers, and options, see *Compaq TCP/IP Services for OpenVMS Management Command Reference*.

17.2.1 Mail Utility Files

Table 17–2 lists the utility files created during the SMTP configuration.

Table 17–2 Default SMTP Utility Files

File Name	Description
LOGIN.COM	Used by the auxiliary server.
TCPIP$SMTP_RECV_RUN.COM	Used by the auxiliary server, and stored in the TCPIP$SYSTEM directory.
TCPIP$SMTP_LOGFILE.LOG	Log of mail queue and symbiont activities.
TCPIP$SMTP_RECV_RUN.LOG	Log of incoming mail.

To analyze the consistency of the SMTP queues against the directories containing the SMTP utility files, enter the ANALYZE MAIL command.

17.2.2 Creating a Postmaster Account

The postmaster account is a required account that receives all undeliverable mail. The SMTP process runs under user account TCPIP$SMTP. Compaq recommends that you do not change this account.

SMTP requires that the system be able to receive mail addressed to the user name POSTMASTER. Set OpenVMS Mail to forward the mail addressed to POSTMASTER to the SYSTEM account. For example:

```
$ SET PROC/PRIV=SYSPRV
$ MAIL
MAIL> SET FORWARD/USER=POSTMASTER SYSTEM
MAIL> SET FORWARD/USER=TCPIP$SMTP SYSTEM
MAIL> SET FORWARD/USER=UCX_SMTP   SYSTEM
```

This ensures that mail messages that could be neither delivered nor bounced back to the sender are sent to the SYSTEM user (usually the system manager).

You can modify the `From:` address on undelivered mail by specifying a user name as the value for the following logical name:

```
$ DEFINE /SYSTEM TCPIP$SMTP_POSTMASTER_ALIAS user-name
```

In this example, *user-name* is the user name without the *domain* portion of the address. For more information, see Section 17.5.

By default, undelivered mail bears the following `From` address:

`TCPIP$SMTP@node.domain`

17.3 Creating a Local Alias File

You can used a local alias to define a list of domains that SMTP will interpret as local. If SMTP receives mail for any of the domains specified as local aliases, it will deliver the mail on the local system using OpenVMS Mail rather than forward it on to another system.

This is useful in an OpenVMS Cluster environment, where you want mail sent to any of the cluster hosts to be delivered locally rather than take the extra step of relaying it from one cluster node to another. It is also useful if you want to set up your OpenVMS host to receive inbound mail either for different domains unrelated to the actual domain of your host or for alias names of your host.

For example, if your host was `a.b.com` and you had entries for `x.y.com` and `y.z.com` in your local alias file, any mail to `x.y.com` and `y.z.com` would be delivered locally on your host. (To implement this fully, set up DNS MX records so that mail to the `x.y.com` and `y.z.com` domains is routed to your host.) For more information about setting up DNS records, see Chapter 5.

To define a list of domains that SMTP interprets as local:

1. Create the file TCPIP$SMTP_LOCAL_ALIASES.TXT.
2. Include a list of domain names that are to be recognized as local. The domain names should have a maximum of 64 characters with one line per name, up to a maximum of 255 names. For example:

```
!
! This is the local alias file.
!
ourdomain.edu
ourdomain1.edu
ourdomain2.edu
ourdomain3.edu
```

3. Copy the TCPIP$SMTP_LOCAL_ALIASES.TXT file to one of the following locations:
   - TCPIP$SMTP_COMMON, where each host listed in the TCPIP$SMTP_LOCAL_ALIASES.TXT file receives clusterwide messages
   - SYS$SPECIFIC:[TCPIP$SMTP] (local system use)
4. Stop and then restart SMTP for the change to take effect.

If SMTP cannot locate the TCPIP$SMTP_LOCAL_ALIASES.TXT file, it looks for the file TCPIP$SMTP_COMMON:UCX$SMTP_LOCAL_ALIASES.TXT. This ensures functionality for mixed clusters (that is, clusters running the current version of TCP/IP Services and earlier versions of the product (UCX)), where the TCPIP$SMTP_COMMON and UCX$SMTP_COMMON logicals point to the same directory. Note that when SMTP looks for UCX$SMTP_LOCAL_ALIASES.TXT it looks for it in the TCPIP$SMTP_COMMON: directory rather than in the UCX$SMTP_COMMON: directory.

17.4 Managing SMTP

Table 17–3 summarizes the commands you use to monitor and manage SMTP.

Table 17–3 SMTP Management Commands

Command	Function	Required Privilege
ANALYZE MAIL	Verifies the consistency of the SMTP queues against the SMTP working directory.	SYSPRV or BYPASS.
DISABLE SERVICE SMTP	Stops SMTP service.	Follows OpenVMS file protection rules.
ENABLE SERVICE SMTP	Initializes communications for SMTP.	Follows OpenVMS file protection rules.
REMOVE MAIL	Deletes the specified mail entries from the SMTP queues.	
SEND MAIL	SMTP requeues a mail message for delivery.	SYSPRV or BYPASS for messages other than yours.
SET CONFIGURATION SMTP	Modifies the characteristics of the SMTP sender and receiver.	SYSPRV or BYPASS.
SHOW CONFIGURATION SMTP	Displays the system characteristics for SMTP.	Follows OpenVMS file protection rules.
SET SERVICE SMTP	Defines, modifies, or deletes the SMTP service in the services database.	SYSPRV or BYPASS.

(continued on next page)

Table 17–3 (Cont.) SMTP Management Commands

Command	Function	Required Privilege
SHOW MAIL	Displays information about mail for the specified user.	SYSPRV or BYPASS.
SHOW SERVICE SMTP	Displays statistical information about the SMTP server.	Follows OpenVMS file protection rules.
START MAIL	Starts the SMTP queuing mechanism.	SYSPRV or BYPASS.
STOP MAIL	Stops the SMTP queuing mechanism.	SYSPRV or BYPASS.

17.4.1 Displaying Mail Queues

To monitor the mail queues, examine the TCPIP$SMTP_LOGFILE.LOG and the TCPIP$SMTP_RECV_RUN.LOG files.

17.4.2 Changing the Number of Mail Queues

To change the number of SMTP queues, follow these steps:

1. Stop SMTP and MAIL on the root node by entering the following commands:

   ```
   TCPIP> DISABLE SERVICE SMTP
   TCPIP> STOP MAIL
   ```

2. Change the SMTP configuration by entering the following command:

   ```
   TCPIP> SET CONFIGURATION SMTP/QUEUES=new_number
   ```

   The maximum number of queues set with this command is 10.

3. Restart SMTP and MAIL by entering the following commands:

   ```
   TCPIP> START MAIL
   TCPIP> ENABLE SERVICE SMTP
   ```

17.4.3 Displaying SMTP Routing Information

To display SMTP routing information, use the SHOW MX_RECORDS command. If you omit *destination* from the command line, you see the entries in the local MX database.

If you specify *destination*, you see all the entries in all the databases that the SMTP mailer would look at, if necessary, to route mail to the destination. The local MX database and the DNS MX database are usually as far as TCP/IP Services needs to search.

17.4.4 SMTP Logging

SMTP logs mail queue and mail symbiont events to the following files:

- TCPIP$SMTP_LOGFILE.LOG
- TCPIP$SMTP_RECV_RUN.LOG

The symbiont and receiver contain a feature called **snapshot logging**, which allows you to run with full diagnostics enabled but to write the diagnostics to the log file only if an error is signaled. This feature saves disk space and allows the receiver or the symbiont, or both, to run at a normal speed. As each line of diagnostic text is generated, it is saved in an internal snapshot buffer rather than to the disk. The buffer is circular in that once it fills up, new lines of text start

to overwrite the old data already there. This functionality provides a snapshot of the last lines of diagnostic text.

Logical names are available to modify the way SMTP logs information and the type of information it reports. These are described in Section 17.5.

17.4.5 Starting and Stopping SMTP

SMTP consists of two components: the sender (the queuing mechanism) and the receiver. You must start the sender before enabling the receiver. The receiver is activated by the auxiliary server.

The SMTP can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$SMTP_STARTUP.COM allows you to start up the SMTP independently.
- SYS$STARTUP:TCPIP$SMTP_SHUTDOWN.COM allows you to shut down the SMTP independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$SMTP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the SMTP is started.
- SYS$STARTUP:TCPIP$SMTP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the SMTP is shut down.

The SMTP services can be started automatically using the TCPIP$CONFIG configuration procedure, or manually using the following command:

```
$ @SYS$STARTUP:TCPIP$SMTP_STARTUP.COM
```

To stop SMTP, enter:

```
$ @SYS$STARTUP:TCPIP$SMTP_SHUTDOWN.COM
```

17.5 Modifying the SMTP Configuration

You can modify the SMTP configuration by defining logical names that are translated at queue startup time. Characteristics you can control include:

- Event-and error-logging diagnostics
- How mail headers are displayed
- How mail is routed
- How SMTP interacts with OpenVMS Mail

Some SMTP logical names are used to either enable or disable a configuration option and do not require a value. If you define the logical name, the option is considered to be enabled. If not defined, the option is disabled. To disable an option, remove the logical name. By convention you should define these logicals to a value of 1.

For example, to enable message logging for messages received from SMTP clients, define the TCPIP$SMTP_RECV_TRACE as follows:

```
$ DEFINE/SYSTEM TCPIP$SMTP_RECV_TRACE 1
```

Other logical names require that you supply a value. For example, to enable logging that provides information about symbiont activity during control file processing, define the logical name TCPIP$SMTP_LOG_LEVEL with a value of 3. For example:

```
$ DEFINE/SYSTEM TCPIP$SMTP_LOG_LEVEL 3
```

When you redefine the value of a logical, you must restart SMTP using the STOP MAIL and START MAIL commands. Use the following files to store logical settings:

- SYS$STARTUP:TCPIP$SMTP_SYSTARTUP.COM
- SYS$STARTUP:TCPIP$SMTP_SYSHUTDOWN.COM

The following descriptions indicate where a value is required.

Note

Define the SMTP configuration logical names as /SYSTEM except where noted.

- TCPIP$SMTP_LOG_LEVEL *value*

  Writes diagnostic information to the log file. Valid numeric values are:

  Value 2 Enables logging of all information when the symbiont starts up.

  The `Next Open File` message is printed, giving the name of each control file before processing begins. All mail headers and mail recipients in a control file are logged after control file processing is complete.

  Value 3 Provides additional information about symbiont initialization and activity during control file processing.

  Value 5 Enables full symbiont diagnostics. For use only under the advice of Compaq's customer support.

- TCPIP$SMTP_NOSEY

  Used with TCPIP$SMTP_LOG_LEVEL to print the full subject RFC headers information. If not defined, the header is logged as `SUBJECT:<omitted>`.

- TCPIP$SMTP_LOG_LINE_NUMBERS

  Writes line numbers to SMTP logs. Includes the symbiont, receiver, and MAIL$PROTOCOL (DEBUG.TXT) logs.

- TCPIP$SMTP_SYMB_TRACE

  Logs all messages received from and transmitted to remote SMTP servers. Used to trace the SMTP application layer protocol. Any nonprinting characters or control characters that are sent or received are printed as *n*, where *n* is the hexadecimal value of the character. For example, command lines and replies are terminated with a `<CR><LF>` that appear in the log file as follows:

  ```
  send buf=MAIL FROM:<jones@acme.com>\d\a
  recv buf=250 <jones@acme.com>... Sender OK\d\a
  ```

In this message, \ *d* \ *a* is the <CR><LF>.

- TCPIP$SMTP_RECV_TRACE

  Logs all messages received from and transmitted to remote SMTP clients. Used to trace the SMTP application layer protocol. The same conventions for logging nonprinting characters or control characters are used. The logical name UCX$SMTP_PROTO_TRACE is obsolete.

- TCPIP$SMTP_RECV_DEBUG

  Logs full diagnostics, similar to the TCPIP$SMTP_LOG_LEVEL 5 logical name. Obsoletes the logical name TCPIP$SMTP_PROTO_DEBUG.

- TCPIP$SMTP_VMSMAIL_SEND

  Instructs the MAIL$PROTOCOL process to log diagnostics to a file named DEBUG.TXT in the default directory. (Used primarily by Compaq.)

- TCPIP$SMTP_VMSMAIL_PARSE

  Causes the SMTP address parsing code to log diagnostics. The symbiont, receiver, and MAIL$PROTOCOL code are sensitive to this logical name, which is used primarily by Compaq.

- TCPIP$SMTP_SYMB_SNAPSHOT_BLOCKS *n*

  Enables snapshot logging for the symbiont. The value you assign (*n*) specifies the size of the snapshot buffer in OpenVMS blocks (1 block being 512 bytes). Use with TCPIP$SMTP_LOG_LEVEL.

  When you enable snapshot buffering for the symbiont, it takes some time for the symbiont process to stop when you enter a TCPIP STOP MAIL command or you stop the queue. The delay depends on the size of the snapshot buffer and the speed of the system and its disks.

  For example, the following command lines set the log level to 5 and enable snapshot logging for the SMTP symbiont with a snapshot buffer of 200 blocks:

  ```
  $ DEFINE/SYSTEM TCPIP$SMTP_LOG_LEVEL 5
  $ DEFINE/SYSTEM TCPIP$SMTP_SYMB_SNAPSHOT_BLOCKS 200
  ```

- TCPIP$SMTP_RECV_SNAPSHOT_BLOCKS *n*

  Enables snapshot logging for the receiver. The value you assign (*n*) specifies the size of the snapshot buffer in OpenVMS blocks (1 block being 512 bytes). Use with TCPIP$SMTP_LOG_LEVEL.

  For example, the following command line sets all of the receiver diagnostics on and enables snapshot logging for the receiver with a snapshot buffer of 200 blocks:

  ```
  $ DEFINE/SYSTEM TCPIP$SMTP_RECV_DEBUG 1
  $ DEFINE/SYSTEM TCPIP$SMTP_RECV_TRACE 1
  $ DEFINE/SYSTEM TCPIP$SMTP_RECV_SNAPSHOT_BLOCKS 200
  ```

- TCPIP$SMTP_NO_SUBS_DOMAIN_INBOUND

  Instructs SMTP not to consider mail that is sent to the substitute domain as local mail.

  By default, SMTP recognizes mail that is addressed to the substitute domain as local mail. To change this default:

  1. Define the system logical name TCPIP$SMTP_NO_SUBS_DOMAIN_INBOUND.

2. Stop and start the SMTP mail queue using the STOP MAIL and START MAIL commands.

- TCPIP$SMTP_COMMON *common-directory*

  Specifies the default cluster common directory. By default, SMTP looks for distribution list (.DIS) and local alias (TCPIP$SMTP_LOCAL_ALIASES.TXT) files in the SYS$SPECIFIC:[TCPIP$SMTP].

  You must:

  – Define this logical name before SMTP startup.

  – Create the directory with read (R) and write (W) access. If the directory is shared between a system running a previous version of the product (UCX) and this version, granting G:RWE privilege is sufficient, because the UCX_SMTP and TCPIP$SMTP accounts are in the same group.

  – Move the .DIS or .TXT file, or both, to the directory.

  You can also use this logical name as a search list. For example, you might want SMTP to look at the clusterwide directory and then the SYS$SPECIFIC:[TCPIP$SMTP] directory, as follows:

```
$ DEFINE/SYSTEM TCPIP$SMTP_COMMON WORKDISK:[SMTP_DIS], -
_$ SYS$SPECIFIC:[TCPIP$SMTP]
```

  Note that other files such as control files and log files reside in the SYS$SPECIFIC:[TCPIP$SMTP] directory.

- TCPIP$SMTP_JACKET_LOCAL

  Instructs the symbiont to put the SMTP jacket on local-to-local mail to provide sufficient information to the POP server.

- TCPIP$SMTP_INBOUND_NOXVMS

  Instructs the symbiont not to use the RFC X-VMS-To header as the text of the OpenVMS Mail `To:` header and the X-VMS-CC header as the text of the `CC:` line. Instead, the RFC `To:` and `CC:` headers are used. If the TCPIP$SMTP_INBOUND_NOXVMS option is not defined, the SMTP symbiont uses the text of the X-VMS-To and X-VMS-CC headers for the mail header lines.

  SMTP accepts inbound mail from a non-SMTP user for a user who has forwarded mail through SMTP on the local system.

  SMTP sets the OpenVMS Mail `CC:` header.

  If an X-VMS-To or X-VMS-Cc RFC header is not present, SMTP puts the header in the OpenVMS Mail `To:` header.

  If you do not want to put these headers in the OpenVMS Mail `To:` and `CC:` headers:

  1. Define the system logical name TCPIP$SMTP_INBOUND_NOXVMS.

  2. Stop and start the SMTP execution queue using the STOP MAIL and START MAIL commands.

- TCPIP$SMTP_VMSDEF_TO

  Instructs the symbiont not to pass any text for the `To:` line to OpenVMS callable mail when delivering local mail. This option causes OpenVMS callable mail to use the default text for the `To:` line (the user name). Overrides the TCPIP$SMTP_INBOUND_NOXVMS option for the `To:` field.

- TCPIP$SMTP_MTS_ALLIN1

  Used in older versions of TCP/IP Services. When relaying mail from the SMTP environment to MTS (the message router), the symbiont puts `TCPIP$SMTP` into the `From:` field. Otherwise, older versions of MR/MRGATE send the mail back with a `Return path too complicated` error. No longer needed if you are running MR and MRGATE Versions 3.3A.

- TCPIP$SMTP_POSTMASTER_ALIAS *user-name*

  Enables mail bounced by the local host to appear to be from a user name other than TCPIP$SMTP@*node.domain*. Specify the user name portion of the address, not including the host name. For example:

  ```
  $ DEFINE/SYSTEM TCPIP$SMTP_POSTMASTER_ALIAS "Postmaster"
  ```

  In this example, bounced mail sent from the local host appears to be from Postmaster@*node.domain* rather than from TCPIP$SMTP@*node.domain*.

  Be sure to set up a forwarding entry for the user name you specify (see Section 17.2.2).

- TCPIP$SMTP_REWRITE_MTS_FROM

  If you have most or all of your users' mail forwarded to ALL-IN-1, use this logical name to instruct the symbiont to parse the user name out of the complex MTS address and append the local host name instead. As a result, only a simple address is sent to the Internet and any replies are relayed correctly to MTS.

- TCPIP$SMTP_ALTGATE_ALWAYS

  Instructs the symbiont to send all mail that is destined for another system (nonlocal mail) to the alternate gateway. Zone check is not performed.

- TCPIP$SMTP_MX_IF_NOALTGATE

  Instructs the symbiont to use MX records to connect to a host if the alternate gateway cannot be reached.

- TCPIP$SMTP_NO_MX

  Instructs the SMTP symbiont not to use MX records to route mail. Attempts to translate the domain part of each SMTP address into a host name and send the mail directly to that address. If the host name does not translate to an address, the mail is returned. If the host is not available, the mail is queued again.

- TCPIP$SMTP_LOCAL_ALIAS_ONLY

  Instructs the SMTP symbiont to use only the contents of the local alias file for determining whether a mail message is local.

- TCPIP$SMTP_PROHIBIT_USER_HEADERS

  Disables outbound alias processing. This prevents the use of the TCPIP$SMTP_FROM logical.

- TCPIP$SMTP_SFF_REQUIRES_PRIV

  This Boolean logical, if defined, requires users to set either SYSPRV, BYPASS or OPER privileges before using the Send From File (SFF) feature. See Section 17.7 for more information about this feature.

- TCPIP$SMTP_MIME_HACK

  When set, SMTP accepts 8BITMIME requests from SMTP clients, preventing the clients from converting the message into a 7-bit format. For more information, see Section 17.9.2.

17.6 Configuring SMTP AntiSPAM

SPAM is the Internet equivalent of junk mail and is a growing source of annoyance to Internet users. TCP/IP Services SMTP contains antiSPAM, which is designed to inhibit the transmission of SPAM.

SMTP antiSPAM is implemented in the SMTP receiver which, for the purposes of this discussion, is called the SMTP server. The following sections describe how to enable and configure SMTP antiSPAM.

17.6.1 Enabling and Managing SMTP AntiSPAM

To enable and manage SMTP antiSPAM, create or edit the following file:

```
TCPIP$SMTP_COMMON:SMTP.CONFIG
```

The logical name TCPIP$SMTP_COMMON is defined at TCP/IP Services startup (see Section 17.5).

The SMTP.CONFIG file should be owned by TCPIP$SMTP and protection should be set to (W:RE).

SMTP.CONFIG is an ASCII text file consisting of one or more lines formatted as follows:

```
Field1: Value1
Field2: Value2
   .
   .
   .
```

In this format:

- Field names start in column 1, are terminated with a colon (:), and are not case sensitive.
- Values vary depending on the field. Limitations and restrictions are described in Table 17–4.

  If a value consists of a list of items, specify them on multiple lines by pressing the Tab key before continuing the value on the subsequent lines. For example:

  ```
  Field1: Item1,
  [Tab]Item2,
  [Tab]Item3
  Field2: Value2
  ```

  Or specify each value as a separate instance of same field. For example:

  ```
  Field1: Item1
  Field1: Item2
  Field1: Item3
  ```

  An alternative format is:

  ```
  Field1: Item1, Item2, Item3
  ```

The maximum number of characters in a value is 500 characters. Unless otherwise noted, a field's value is not case sensitive.

Fields described as Boolean have the following legal values:

To turn the feature on:	To turn the feature off:
ON	OFF
TRUE	FALSE
1	0
YES	NO

To comment out a line, enter an exclamation point (!) in column 1.

The file SMTP_CONFIG.TEMPLATE is provided to help you create this file; it contains guidelines on configuring antiSPAM.

17.6.1.1 SMTP AntiSPAM Field Names

Table 17–4 describes the field names and values for antiSPAM configuration. .

Table 17–4 AntiSPAM Configuration Options

Field Name	Value	Default
Good-Clients	A list of the IP addresses, IP nets, DNS hostnames, and DNS MX domains of known good SMTP clients.	If not defined, SMTP will not check IP address of SMTP client against this list.
Bad-Clients	A list of the IP addresses, IP nets, DNS hostnames, and DNS MX domains of known bad SMTP clients.	If not defined, SMTP will not check IP address of SMTP client against this list.
Relay-Zones	A list of the SMTP domains to which the system will relay mail even if it is from an unknown client.	If not defined, SMTP will not check recipient address of mail against this list.
RBLs	A list of domains that maintain RBL lists.	If not defined, SMTP will not check IP address of SMTP client against any RBL lists.
Relay-Based-On-Mx	TRUE or FALSE. If TRUE, the SMTP server accepts relays from unknown clients to recipients where the recipient's domain has an MX record naming the local host as a gateway.	FALSE

(continued on next page)

Table 17–4 (Cont.) AntiSPAM Configuration Options

Field Name	Value	Default
Reject-Unbacktranslatable-IP	TRUE or FALSE. If TRUE, the SMTP server rejects any mail from an SMTP client whose IP address cannot be backtranslated to a hostname.	FALSE
Accept-Unqualified-Senders	TRUE or FALSE. If TRUE, the SMTP server accepts mail for which the sender address (the address from the MAIL FROM command) has no domain or an unqualified domain.	FALSE
Accept-Unresolvable-Domains	TRUE or FALSE. If TRUE, the SMTP server accepts mail for which the sender address (the address from the MAIL FROM command) has a domain that cannot be resolved using MX lookup.	FALSE
Reject-Mail-From	A list of wildcarded patterns that are matched against the sender address. If a match occurs, the MAIL FROM command is rejected and the link is disconnected.	If not defined, SMTP will not check the sender address of the mail against the list.
Accept-Mail-From	A list of wildcarded patterns that are matched against the sender address if the sender address has matched one of the entries in the Reject-Mail-From list. If the sender address matches the Accept-Mail-From list, the message is sent on.	If not defined, SMTP will not check the sender address of the mail against the list.
SPAM-Action	Allows you to configure the way SMTP reports a SPAM event. Specify a comma-separated list including one or more of the following: • NONE • OPCOM • ACCOUNTING	OPCOM

(continued on next page)

Table 17–4 (Cont.) AntiSPAM Configuration Options

Field Name	Value	Default
Security	FRIENDLY or SECURE. This value specifies the type of error text sent to the SMTP client when disconnecting a link because of a SPAM event. A value of SECURE means to send purposely unhelpful error text. A value of FRIENDLY means to send helpful error text.	SECURE
Unbacktranslatable-IP-Text Bad-Clients-Text Client-In-RBL-Text Reject-Mail-From-Text Unqualified-Sender-Text Unresolvable-Domain-Text SPAM-Relay-Text	These individual fields (one for each type of SPAM event) hold the error text to be sent to the SMTP client. These override values set in the Security field.	The default for each of these is set according to the value of the Security field. See Section 17.6.8.3 for more information.

The following sections provide further information about the configuration options.

17.6.2 Preventing the System from Routing SPAM

SPAM mailing lists contain thousands of addresses and sending a SPAM takes a great deal of time. Therefore, SPAMmers prefer to use hosts other than their own to send the message. SPAMmers routinely use unaware Internet hosts as route-through hosts for their SPAM. The victim is a host not protected by a firewall or by SPAM-aware software. The SPAMming SMTP client software connects to the victim SMTP server host and issues multiple RCPT TO commands, which may number in the thousands. The SPAMing SMTP client then sends the message to the victim host and closes the link. It is now left to the victim host to do the real work of relaying the SPAM to the thousands of recipients.

Fortunately, the route-through attack can often be detected. Most or all of the recipients of the SPAM will not be within the victim's own domains or IP networks. They will be somewhere outside in the expanse of the Internet. You must trap for the situation where an unknown SMTP client is trying to use your system to relay mail to recipients in domains outside its own. If you specify the "known world" and the "unknown world," the SMTP server can detect this type of SPAM attack.

SMTP allows you to configure two lists:

- Good-Clients, a list of the IP addresses, IP nets, DNS hostnames and DNS MX domains of known good SMTP clients.
- Relay-Zones, a list of the SMTP domains to which SMTP will relay mail even if it is from an unknown client.

Together, these lists define the "known good world" to the SMTP server for relay purposes. They are used to prevent SPAM routing as follows:

1. The SMTP server checks the IP address of the client against the Good-Clients list. If a match occurs, the client is considered "known good" and it is free to use the local system to relay without further checking. However, if no match

against the Good-Clients list occurs, the client is considered "unknown" and the process goes to step 2.

2. When the client is unknown, the domain of the address in each RCPT TO command is checked against the Relay-Zones list. If a match occurs, the RCPT TO command is accepted, because it is a relay from the unknown world to the known world (for example, e-mail from the Internet). If a match does not occur, the RCPT TO is considered unacceptable for route-through.

17.6.3 Controlling Relay Checking

If neither Good-Clients nor Relay-Zones is configured, relay checking depends on the setting of the SMTP configuration relay flag. If the relay flag is set, all relays are allowed; if it is not set, relays are not allowed.

To use Good-Clients and Relay-Zones lists, you must still set the SMTP configuration relay flag. Use the following command:

```
TCPIP> SMTP SET CONFIGURATION/OPTION=RELAY
```

Although the configuration of both Good-Clients and Relay-Zones at the same time makes sense for most configurations, it is not required that both be configured. If a Relay-Zones list is specified without a Good-Clients list, relay checking depends on the setting of the relay flag. Information in the Relay-Zones list is ignored.

If you specify a Good-Clients list but no Relay-Zones list, the host will relay mail from SMTP clients matching the Good-Clients list but will not relay any mail from unknown SMTP client hosts. The host will accept mail only from an unknown SMTP client host if it is local mail (mail destined to the local host or a host listed in the local aliases list). If you want the host to accept relay mail from unknown SMTP client hosts, you must specify Relay-Zones or enable Relay-Based-On-Mx (described in Section 17.6.3.3).

17.6.3.1 Specifying the Good-Clients List

The Good-Clients list is a comma-separated list of clients, specified as one of the following:

- IP address
- IP network
- DNS hostname
- DNS MX domain

To enter an IP network, use the *n.n.n.n/m* syntax, where *n.n.n.n* is the IP network and *m* is the number of bits in the subnet mask. For example:

```
Good-Clients: 1.2.0.0/16, 2.3.0.0/16,
              2.3.4.5, relay.abc.com
```

This Good-Clients list contains two IP networks (1.2.0.0 and 2.3.0.0), an IP address (2.3.4.5), and a DNS entry (`relay.abc.com`). An entry that does not follow the standard IP address or network format is assumed to be a DNS entry.

17.6.3.2 Processing DNS Entries in the Good-Clients List

The SMTP server uses the Good-Clients list to match the IP addresses of SMTP clients. Therefore, entries are stored internally as IP addresses. DNS hostname and MX domain entries are stored as IP addresses, determined by the following process:

1. An entry that is not apparently an IP address or IP network is assumed to be a DNS host name, and the matching IP address is stored in the list.
2. For an entry that cannot be resolved as a DNS host name, the SMTP server looks for MX records.

For configurations where the generic mail server name does not have an associated DNS host name, the SMTP server uses the MX records, which specify mail relay hosts. The following example demonstrates this configuration:

```
TCPIP> show host relay.abc.com
%TCPIP-W-NORECORD, information not found
-RMS-E-RNF, record not found

TCPIP> show mx relay.abc.com

                          BIND MX database

Server:          1.2.3.4          host.abc.com

Gate address     Preference       Gate name

1.3.4.5          100              mail11.abc.com
1.3.5.6          100              mail13.abc.com
2.4.5.6          200              mail2.abc.com
2.4.5.7          200              mail1.abc.com
3.4.5.6          300              mail21.abc.com
3.4.6.7          300              mail12.abc.com
```

To include the addresses listed as MX gateways in this example, enter `relay.abc.com` in the Good-Clients list.

17.6.3.3 Mail Relay to MX Gateways

You can configure the SMTP server to relay mail from an unknown SMTP client to a domain that does not match the entries Relay-Zones but that has an MX record naming the local host as an MX gateway. To enable this feature, set the Relay-Based-On-Mx option to TRUE in SMTP.CONFIG.

For example, the Relay-Zones list is not specified on example host `VMShost.abc.com`. When an unknown host tries to relay mail to `podunk.def.com` through VMShost, and the Relay-Based-On-Mx option is enabled, the SMTP server on VMShost searches for MX records for `podunk.def.com`. If one of PODUNK's MX records lists VMShost as the MX gateway, the relay is accepted, even though the SMTP client is unknown and the RCTP TO address did not match the Relay-Zones list.

17.6.3.4 Specifying the Relay-Zones List

The Relay-Zones list specifies the domains to which the SMTP server will relay mail from unknown SMTP clients. Do not use wildcards in the entries in this list; wildcarding is implicit (that is, `*.domain` is implied). For example:

```
Relay-Zones: def.com,
        abc.com,
        company.com
```

This example specifies the relay of mail from unknown SMTP clients to any host within the def.com, abc.com, or company.com domain. Because of implied wildcarding, domains like VMShost.abc.com match against this list.

17.6.3.5 Rejecting Route-Through Attempts

If the SMTP server does not resolve the a route-through attempt using the Good-Clients list and the Relay-Zones list, it rejects the RCPT TO command. It allows an SMTP client to attempt route-through twice; if a third attempt is made, the SMTP server rejects the RCPT TO command, disconnects the link, and reports a SPAM event. For more information about SPAM event reporting, see Section 17.6.8.

17.6.3.6 Examples of Specifying Good-Clients and Relay-Zones

In the following examples, host.abc.com is the host, and Good-Clients and Relay-Zones lists are configured as follows:

```
Good-Clients: 1.2.0.0/16, 2.3.0.0/16, relay.abc.com
Relay-Zones:  def.com, abc.com, company.com
```

The Good-Clients list specifies clients whose IP addresses are in the 1.2 or 2.3 subnets or whose IP addresses match the relay.abc.com.

The following examples assume that host.abc.com is not protected by a firewall and has direct Internet connectivity.

1. The following example explains the process of handling a mail message where the client is unknown and RCPT TO address is unknown.

   A host with the IP address 2.2.3.5 connects to VMShost's SMTP server. The client sends a RCPT TO address of jones@someplace.else.com. The SMTP server:

   a. Fails to find a matching IP address in the Good-Clients list. The client is considered unknown.

   b. Fails to find the domain of the RCPT TO address in the Relay-Zones list.

   c. The RCPT TO command is rejected with the following message:

      ```
      <<<RCPT TO:<jones@someplace.else.com>
      >>>550 User not local, Relay disabled.
      ```

2. This example shows the process of handling a mail message for which the client is unknown but the RCPT TO address is accepted.

   A host with the IP address 2.2.3.5 connects to VMShost's SMTP server. This IP address does not match Good-Clients, so the client is considered unknown.

   However, if the client sends a RCPT TO address of smith@foobar.xxx.def.com, the domain of the RCPT TO address is matched against the Relay-Zones list. The RCPT TO address foobar.xxx.def.com matches the Relay-Zones list, so the RCPT TO command is accepted.

3. In this example, the client with IP address 1.2.1.2 connects to VMShost's SMTP server. This IP address matches Good-Clients (it is in subnet 1.2). Therefore, the client is considered known. The SMTP server does not check the domains of the RCPT TO addresses.

17.6.4 Blocking Mail from Specified Clients

You can configure the SMTP server to automatically reject any mail transactions with specified SMTP clients. To enable this feature, configure the Bad-Clients list in SMTP.CONFIG. The syntax of the Bad-Clients list is the same as the Good-Clients list. For example:

```
Bad-Clients: 1.2.3.5, 100.101.102.103
```

If Bad-Clients is configured, the SMTP server checks the IP address of the client against the list. If a match occurs, the SMTP client is considered "known bad;" the server sends a failure message to the client and then disconnects the link.

17.6.4.1 Resolving Conflicts between Bad-Clients and Good-Clients

The Bad-Clients and Good-Clients lists are not mutually exclusive. If an SMTP client's IP address may be resolved in both lists, the entry that most closely matches the client's IP address is used.

For example, the following lists are configured:

```
Bad-Clients: 1.0.0.0/8
Good-Clients: 1.2.3.6
```

When an SMTP connection comes in from IP address 1.2.3.6, which is in the 1.0.0.0 subnet, the client may be considered a known bad client. But because the specific IP address is specified in the Good-Clients list, the message is accepted.

In a case where the exact IP address is specified in both the Good-Clients list and the Bad-Clients list, the SMTP server accepts the message.

17.6.5 Real-Time Black Hole Lists (RBL)

To prevent SPAM route-through from clients dialed through an ISP and connecting directly to the destination hosts, the Internet community maintains a list of known SPAMming IP addresses at domain `rbl.maps.vix.com`. This is called the Realtime Blackhole List (RBL) and contains DNS A records.

To determine whether a specific IP address is in the RBL list, perform these steps:

1. Reverse the IP address.
2. Append the domain of the RBL.
3. Do a DNS A record lookup.

For example, to check the IP address of 2.2.3.5, do a name lookup of `5.3.2.2.rbl.maps.vix.com`. If the query returns something, then IP address 2.2.3.5 is in the `rbl.maps.vix.com` SPAMmer list. All RBLs are implemented this way.

To take advantage of the RBL, configure the RBLs list in SMTP.CONFIG. The RBLs configuration option lists the domains providing RBL services. You can specify a list of RBLs, thereby accommodating individual RBLs and additional Internet-provided RBLs along with the current one.

For example:

```
RBLs: rbl.maps.vix.com, rbl.ourcompany.com
```

If the SMTP server matches the IP address of the client with an entry in any of the RBLs in the list, the server sends a failure message to the client and disconnects the link.

If a client IP address matches one in the Good-Clients list, the message is accepted; the SMTP server does not check the RBLs.

17.6.5.1 Using Other RBL Lists

Other lists serve similar functions to the RBL list. For example:

- MAPS Dial-up User List — A list of IP addresses that participating ISPs have allocated to them.

  If you want to include a check against this list, add `dul.maps.vix.com` to the RBL list.

  Even when SMTP is set up to reject all incoming SMTP connections from dial-up IP addresses, mail can still be received from someone who is dialed up to an Internet Service Provider (ISP). This is because they send their SMTP mail to their ISP's mail servers, and the ISP mail server connects to your mail server. The IP address of the ISP's mail server is not in an RBL, so this connection will not be rejected.

  This RBL list only prevents routing of e-mail from a dialup IP address directly to your host.

  You can allow your own dialup users to connect directly to your hosts without being rejected. If your dial-up addresses have been included in an RBL, you can exclude them from the RBL check by adding the IP addresses to the Good-Clients list.

  For more information about MAPS list, visit the following web site:

  `http://maps.vix.com`

- Open Relay Behavior modification (ORBS) list — A list of SMTP servers that are known to permit third-party relay.

  To include this list, add `relays.orbs.org` to the RBLs list in SMTP.CONFIG.

  Caution

  If you include `relays.orbs.org` in the RBLs list, you will not receive mail from any host IP address in the ORBS list. This could prevent legitimate mail from coming in.

  For more information about the ORBS list, visit the following web site:

  `http://www.orbs.org`

17.6.6 Translating Client IP Addresses

You can configure SMTP to translate the client's IP address to a host name, and to disconnect the link if no host name exists. To enable this feature, set the Reject-Unbacktranslatable-IP option in SMTP.CONFIG. Translation is not performed if the SMTP client's IP address matches an entry in the Good-Clients list.

17.6.7 Blocking Mail from Specified Senders

You configure SMTP to reject mail based on the address of the sender. The sender's address is specified in the MAIL FROM command. (The terms "sender address" and "MAIL FROM address" are synonymous.) To specify sender addresses from whom mail will always be rejected, include the Reject-Mail-From list in the SMTP.CONFIG file.

The Reject-Mail-From list includes wildcarded patterns that are checked against the sender address. If the SMTP server matches the sender address against a pattern in the Reject-Mail_From list, the MAIL FROM command is rejected and the link is disconnected. Wildcarded patterns may include the standard asterisk (*) and percent sign (%) wildcard characters.

For example:

```
Reject-Mail-From: *.xyz.com, known.spammer@*, *the_internet*
```

To specify hosts from which to allow mail, even if the address matches that specified in the Reject-Mail-From list, include them in the Accept-Mail-From list in SMTP.CONFIG.

The Accept-Mail-From list includes wildcarded patterns that are checked against the sender address. If the SMTP server finds that the MAIL FROM address matches an entry in the Reject-Mail-From list, it then checks the Accept-Mail-From list also. You can use this list to allow mail from legitimate senders in the domains listed in the Reject-Mail-From list.

For example:

```
Accept-Mail-From: *@notabadguy.xyz.com, the_internet_news@somehwere.com
```

In this example, the entry `the_internet_news@somehwere.com` allows mail from the sender address `the_internet_news@somehwere.com`, even though it matches the entry `*the_internet*` from the Reject-Mail-From list. Likewise, it accepts mail from `jones@notabadguy.xyz.com`, even though it matches the entry `*.xyz.com` in the Reject-Mail-From list.

In addition to the Accept-Mail-From list, you can specify the following configuration options in SMTP.CONFIG to allow mail from senders in the Reject-Mail-From list:

- Accept-Unqualified-Senders

  By default, if the TCP/IP Services SMTP server receives a message with an unqualified sender address, or with a sender address with no domain at all, it will reject the MAIL FROM command and disconnect the link.

  For example, the following sender addresses would be rejected by default:

  ```
  MAIL FROM:<somebody>
  MAIL FROM:<somebody@someplace>
  ```

  The first address has no domain and the second has an unqualified domain.

  To accept mail with these types of sender addresses, set Accept-Unqualified-Senders in SMTP.CONFIG, as follows:

  ```
  Accept-Unqualified-Senders: TRUE
  ```

  When the Accept-Unqualified-Senders option is set, the SMTP server does not check whether the sender address either has a domain or is fully qualified.

- Accept-Unresolvable-Domains

By default, if the SMTP server fails to find a MX record for the sender address, it rejects the MAIL FROM command and disconnects the link.

You can specify that messages with unresolvable domains be accepted by setting the Accept-Unresolvable-Domains configuration option to TRUE in SMTP.CONFIG, as follows:

```
Accept-Unresolvable-Domains: TRUE
```

When Accept-Unresolvable-Domains is set, the SMTP server will not perform an MX lookup on the sender address.

17.6.8 Specifying Handling of SPAM Events

Whenever the TCP/IP Services SMTP server disconnects a link with a client, it generates an event message. You can control the way events are handled using the procedures in the following sections.

17.6.8.1 Reporting SPAM Events

You can customize the SMTP server to report a SPAM event in the following ways. The SMTP server can:

- Send an OPCOM message.
- Send a /TYPE=USER message to the accounting subsystem.

To configure the way SMTP reports the event, use the SPAM-Action field in SMTP.CONFIG. The legal values are:

- NONE
- OPCOM (the default)
- ACCOUNTING

You can specify multiple values for the SPAM-Action field. For example:

```
SPAM-Action: OPCOM, ACCOUNTING
```

This example causes both OPCOM and accounting messages to be sent for each SPAM event. To disable SPAM event reporting, enter a value of NONE for SPAM-Action in SMTP.CONFIG, as follows:

```
SPAM-Action: NONE
```

17.6.8.2 Configuring SPAM Security

When the SMTP server disconnects the link with the client because of one of the antiSPAM checks, it sends a message back to the client. The text of the message is controlled by the Security field in SMTP.CONFIG. The legal values for this field are:

- SECURE (the default)

  If Security is set to SECURE, the messages do not indicate the cause of the disconnect.

- FRIENDLY

  If Security is set to FRIENDLY, the messages indicate the cause of the disconnect.

17.6.8.3 Specifying the SPAM Rejection Text

You can specify the rejection text message to be sent to the client. The field names for these options end in "-Text", and the values for them must be a single line of text. These fields override the default text associated with the specific SPAM event.

The following are the fields and default messages for the SECURE option:

- Unbacktranslatable-IP-Text: Closing transmission channel.
- Bad-Clients-Text: Closing transmission channel.
- Client-In-RBL-Text: Closing transmission channel.
- Reject-Mail-From-Text: Closing transmission channel.
- Unqualified-Sender-Text: Closing transmission channel.
- Unresolvable-Domain-Text: Closing transmission channel.
- SPAM-Relay-Text: User not local, Relay disabled.

The following are the fields and default messages for the FRIENDLY option:

- Unbacktranslatable-IP-Text: I can't backtranslate your IP address to a host name.
- Bad-Clients-Text: Your IP address or subnet is in my list of bad ones.
- Client-In-RBL-Text: Your IP address is in my RBL list.
- Reject-Mail-From-Text: That sender address is in my list of bad ones.
- Unqualified-Sender-Text: That sender address is unqualified.
- Unresolvable-Domain-Text: That sender address is unresolvable into a host name or MX domain.
- SPAM-Relay-Text: Both you and the recipient are unknown to me. I will not relay.

You can change one or more of the default messages by including the field and your message for a value. This will override the default setting for that field. For example:

```
Unbacktranslatable-IP-Text: Your IP address is unbacktranslatable. SPAMMER!
```

17.7 Managing SMTP Send-From-File (SFF)

SMTP allows you to create a mail message in a file and send it to the SMTP mailer to be delivered with headers you specify. Using SFF, you can create automated tools that compose and send mail messages.

SFF is also useful for forwarding nontext (MIME) files because it prevents the mailer from encapsulating the MIME and SMTP headers in the body of a new mail message. In this way, SMTP functions like the `redirect` command on your personal computer.

17.7.1 SFF Security Measures

The ability to create messages with arbitrary headers could be used to spoof message headers. To limit this, the SFF mechanism includes the following security measures:

- Allows you to define the TCPIP$SMTP_SFF_REQUIRES_PRIV logical. If the logical is defined as 1, the process must have either BYPASS, SYSPRV, or OPER privilege set in order to use SFF.
- To help you track messages, SFF adds a Received: header to the headers you supply. This tells you the origin of an attempted spoofed message.

You can invoke SFF from an application or from DCL, as described in the following sections.

17.7.2 Invoking SFF from an Application

TCPIP$SMTP_MAILSHR.EXE contains a routine called TCPIP$SMTP_SEND_FROM_FILE. This routine is declared as follows:

```
unsigned int TCPIP$SMTP_SEND_FROM_FILE(infile_name, logfd, log_level)
char *infile_name;
FILE *logfd;
int log_level;
```

The parameters for this routine are:

- *infile_name*

  Specifies the name of the text file that contains the RFC 822 mail message.

- *logfd*

  Specifies the file to which to log diagnostic messages. This file must be opened by the caller before calling this routine. If no log file is specified, output goes to SYS$OUTPUT. This parameter is optional.

- *log_level*

  Specifies the level of diagnostics to use: either 1 (on) or 0 (off). The default is 0 (no logging). This parameter is optional.

To call the routine, link with TCPIP$SMTP_MAILSHR.EXE/SHARE.

17.7.3 Invoking SFF from DCL

The SMTP_SFF command allows you to invoke SFF. To define SMTP_SFF as a foreign command so that you can use it from DCL, enter the following command:

```
$ SMTP_SFF:==$TCPIP$SYSTEM:TCPIP$SMTP_SFF.EXE
```

This command takes UNIX style parameters and passes them to SFF.

The command format is:

```
SMTP_SFF infile_name [-log logfile_name] [-loglevel log_level]
```

The parameters to this command are:

- *infile_name*

  Specifies the name of text input file containing the SMTP mail to send.

- *logfile_name*

  Specifies the name of the log file for diagnostic messages. The default is SYS$OUTPUT. This parameter is optional.

- *log_level*

  Specifies the debug log level: either 1 (on) or 0 (off). The default is 0 (no logging). This parameter is optional.

17.8 Disabling SMTP Outbound Alias

Users can specify an outbound alias that is applied to mail as it is sent and specifies the network address to which a reply will be sent. The outbound alias is defined using the TCPIP$SMTP_FROM logical, as described in Section 17.5.

To disable outbound alias processing (prevent the use of the TCPIP$SMTP_FROM logical), define the following system logical:

```
$ DEFINE/SYSTEM TCPIP$SMTP_PROHIBIT_USER_HEADERS 1
```

17.9 Solving SMTP Problems

To isolate an SMTP problem, follow these steps:

1. Check the directory SYS$SPECIFIC:[TCPIP$SMTP] for the following log files:

   - TCPIP$SMTP_LOGFILE.LOG

     This log file monitors queue activity.

   - TCPIP$SMTP_RECV_LOGFILE.LOG

     This log file is created with every message received.

   Purge the directory regularly.

2. Use the TCPIP$SMTP_LOG_LEVEL logical, as described in Section 17.5.

3. Check the mail in the TCPIP$SMTP account.

   Forward TCPIP$SMTP mail to the SYSTEM account for monitoring. By default, remote login to TCPIP$SMTP is not allowed.

4. Check the directory SYS$SPECIFIC:[TCPIP$SMTP] for lost mail.

   If an incoming mail message was undeliverable and the error message was also undeliverable, the SMTP control file is left in this directory, not in the queue.

5. Check the consistency of the SMTP queues against the directories with the SMTP utility files.

   Enter the ANALYZE MAIL command (see Section 17.9.1).

17.9.1 Verifying SMTP Control Files

Use the ANALYZE MAIL command to verify the correspondence of the SMTP queues with SMTP control files. This command does the following:

- Checks that all the current entries in the SMTP queues have a supporting control file in the mail directory of a user. You can specify a user or analyze the mail of all users.
- Checks that there are no lost control files in the SMTP working directory.
- The /DELETE qualifier deletes each control file lacking a corresponding queue entry.

- The /REPAIR qualifier fixes these errors:
  - Resubmits for delivery each valid control file in the SMTP directory with no entry in an SMTP queue.
  - Deletes each invalid control file (fails the internal consistency check) and the corresponding queue entry.
  - Either requeues or deletes messages placed on hold.

The following examples show how to use the ANALYZE MAIL command:

1. The following command encounters a problem, displays a description and solution, and then requests confirmation before fixing each record.

```
TCPIP> ANALYZE MAIL /REPAIR /CONFIRM

%TCPIP-E-ANA_SUP_BADIICGSIZE, Problem: Bad initial inode cell
group size: bad_value
Solution: Will be replaced by
default size: good_value
        CONFIRM [Y/N/G]:
```

2. The following command creates a summary of SMTP entries and control files for user DRAKE.

```
TCPIP> ANALYZE MAIL DRAKE

%TCPIP-I-ANA_RUNING, ANALYZE runs on node DODO

%TCPIP-I-ANA_NOENTR, no queue entry found for file
NEST3$:[DRAKE]93042311394417_DRAKE.UCX_DODO;1

%TCPIP-I-ANA_COMPLE, ANALYZE completed on node DODO

%TCPIP-I-ANA_FEPAIR, found 0 file-queue entry pairs
%TCPIP-I-ANA_DELQEN, deleted 0 queue entries
%TCPIP-I-ANA_FILNOQ, found 1 files with no queue entries
%TCPIP-I-ANA_FILHLD, holding 0 files in directory
%TCPIP-I-ANA_FILDEL, deleted 0 files from the Postmaster directory
%TCPIP-I-ANA_SUBFIL, submitted 0 files to the generic queue
%TCPIP-I-ANA_FILACE, encountered 0 file access errors
%TCPIP-I-ANA_NONCFF, found 0 non-unknown files in Postmaster directory
%TCPIP-I-ANA_FILCOR, found 0 corrupted CF files in Postmaster directory
```

3. The following command:
   - Creates a summary of SMTP entries and control files for user DRAKE.
   - Requeues control files lacking corresponding queue entries.
   - Deletes control files created before November 24, 1999.

```
TCPIP> ANALYZE MAIL DRAKE /REPAIR /DELETE=BEFORE=24-NOV-1999
```

17.9.2 Preventing 8-Bit Autoconversion

You can prevent the SMTP client from converting 8-bit characters to 7-bit characters before sending the mail message to the SMTP server. On some displays, such as that used by OpenVMS Mail (a character-cell based mailer), certain 8-bit strings, such as accented characters, are converted and displayed in coded sequences.

To prevent this behavior, set the following logical:

```
$ DEFINE/SYS/EXEC TCPIP$SMTP_8BITMIME_HACK 1
```

When set, the SMTP receiver tells SMTP clients that 8-bit characters are supported. In this case, the client does not convert them to 7-bit format.

18

Configuring and Managing the POP Server

The Post Office Protocol (POP) server and the Simple Mail Transfer Protocol (SMTP) server software work together to provide reliable mail management in a client/server environment.

The POP server acts as an interface to the mail repository. It accepts and stores mail messages for you, even when your client system is not connected, and forwards those messages to you at your request. POP is used mostly by PC clients to ensure that mail is received and retained even when the system is not connected to the network.

After the POP server is enabled on your system, you can modify the default characteristics by defining logical names.

This chapter reviews key POP concepts and describes:

- How to start up and shut down the POP server (Section 18.2)
- How to modify POP server characteristics (Section 18.3)
- How to enable MIME mail using POP (Section 18.4)
- How to solve POP problems (Section 18.5)

18.1 Key Concepts

The POP server is an implementation of the Post Office Protocol Version 3 server (the public domain IUPOP3 server) specified in RFC 1725.

The POP server is intended to be used as a mail repository for:

- PC systems that may not be connected to a network for periods of time
- Smaller nodes that may not have sufficient resources to keep an SMTP server and associated local mail delivery system resident and continuously running

With POP, mail is delivered to a shared mail server, and a user periodically downloads unread mail. Once delivered, the messages are deleted from the server.

The POP server is assigned port 110, and all POP client connections are made to this port.

The following sections review the POP process and describe how the TCP/IP Services software implements POP. If you are not familiar with POP, refer to RFC 1725 or introductory POP documentation for more information.

18.1.1 POP Server Process

The POP server is installed with SYSPRV and BYPASS privileges and runs in the TCPIP$POP account, which receives the correct quotas from the TCPIP$CONFIG procedure. The POP server is invoked by the auxiliary server.

The POP server uses security features provided in the protocol and in the OpenVMS operating system, as well as additional security measures. These methods provide a secure process that minimizes the possibility of inappropriate access to a user's mail file on the served system.

You can modify the POP server default characteristics and implement new characteristics by defining the system logical names outlined in Section 18.3.

18.1.2 How to Access Mail Messages from the POP Server

To access mail messages from the POP server, you configure a user name and password, or the POP shared secret-password string, into your client mail application.

Your client system opens the TCP connection and attempts to access the server by entering applicable POP commands such as USER (user name) and PASS (password), or APOP (shared secret password). In addition, POP supports the UID command, which some POP clients use, where the UID (user identification) that POP creates for each mail message is a concatenation of the user name and the date of arrival.

Once your client system opens the TCP connection, the POP server issues the following greeting:

```
+OK  POP server ready TCPIP V5.1 [hostname and IP_Address]
```

By default, the POP server reads mail from the user's OpenVMS NEWMAIL folder. If you do not instruct the POP server to delete the mail, the server either moves the mail to the MAIL folder (if the logical name TCPIP$POP_USE_MAIL_FOLDER is defined) or keeps it in the NEWMAIL folder (if the logical name TCPIP$POP_LEAVE_IN_NEWMAIL is defined). These logical names are described in Section 18.3.

18.1.3 How the POP Server Initiates and Manages a TCP Connection

The POP server starts the service by listening on TCP port 110. The client initiates a connection when it wants to make use of the POP service. The POP server sends either a greeting message confirming the connection (a message with the +OK prefix) or a message that the connection was not successful (a message with the -ERR prefix).

POP permits only two user name and password authorization attempts per TCP connection. After the second failure, POP closes the connection. Once connected, the client and server exchange commands and responses.

When the POP server detects a blocked TCP connection, it suspends output to the connection for 2 seconds to allow it to unblock. Upon retry, if the connection is still blocked, the POP server waits 4 seconds before trying again, and so on up to 32 seconds. If the connection is still blocked after 32 seconds, the POP server shuts down the connection and sends an error message to the log file, allowing other client connections to continue to operate.

18.1.4 How the POP Server Handles Foreign Message Formats

POP contains minimal support for mail messages that contain foreign formats. Such messages are usually binary and therefore are not transferred to the POP client. Instead, the POP server transfers the message headers, along with a brief message instructing the user to log in and extract the foreign message into a file. Foreign messages are moved into your MAIL folder; they are never deleted by the POP server.

18.1.5 How the POP Server Authorizes Users

Table 18–1 outlines the methods the POP server process uses to authorize user access.

Table 18–1 POP User Authorization Methods

Method	Description
Shared secret-password string	Most secure POP server access method. Initiated by the client system through the APOP command. Allows a user to become authorized by the POP server without the need to send a password over the network. Eliminates a potential path for unauthorized users to obtain a password and break into the system. POP requires a shared secret string from any user who wants to read mail using the APOP authorization method. For information about creating the shared secret string, see the *DIGITAL TCP/IP Services for OpenVMS User's Guide*.
User name and password	Least secure POP server access method. Initiated by the client system through the USER and PASS commands. The POP server authorizes the client to access the desired mailbox based on receipt of a valid user name and password. 1. The user configures a user name and password into the POP client system. Each POP client has its own method of configuring. Note that the user name and password pair is the user name and password for the TCP/IP Services system, not for the POP client system. 2. The POP client sends the user name and password pair to the server, and the server confirms the pair against that in the OpenVMS SYSUAF file. Note that the password is sent unencrypted over the TCP connection, which might cause security problems for some environments. Upon authorization, the POP server allows access to the user's OpenVMS mailbox.
OpenVMS SYSUAF settings on user accounts	Access to the POP server is not permitted if: • Either the DISMAIL or DISUSER flags are set for the account. • The account has expired according to the SYSUAF expiration date. • Access has been denied because of an incorrect user name and password.
Ability to disable the USER and PASS commands	Allows the system manager to use the APOP authorization method for all POP clients, the more secure means of user authorization. When you disable the USER and PASS commands (by defining the logical name TCPIP$POP_DISUSERPASS), the POP server responds to the commands with a failure message.

18.1.6 Understanding POP Message Headers

Mail message headers sent by the POP server must conform to the standard specified for SMTP in RFC 822. Because many of the messages received on an OpenVMS system are not in the SMTP format (for example, DECnet mail or mail from another message transport system), the POP server builds a new set of headers for each message based on the OpenVMS message headers.

The headers on mail messages forwarded by the POP server are as follows:

POP Message Header	Obtained From
`Date:`	Arrival date of message. Changed to UNIX format.
`From:`	OpenVMS message `From:` field. Rebuilt to ensure RFC 822 compatibility. See Section 18.1.6.1.
`To:`	OpenVMS Mail `To:` field. Not rebuilt.
`CC:`	OpenVMS Mail `CC:` field. Not rebuilt.
`Subject:`	OpenVMS Mail `Subj:` field. Not rebuilt.
`X-VMS-From:`	OpenVMS Mail `From:` field. Not rebuilt.
`X-POP3-Server:`	Server host name and POP version information. Sent only if logical name TCPIP$POP_SEND_ID_HEADERS is defined.
`X-POP3-ID:`	Message UID. Sent only if logical name TCPIP$POP_SEND_ID_HEADERS is defined.

The POP server sends these message headers to the POP client unless all of the following conditions are true:

- The TCPIP$POP_IGNORE_MAIL11_HEADERS logical name is defined (see Section 18.3).
- The `From:` address is an SMTP address.
- The SMTP qualifier /OPTION=TOP_HEADERS is set.

Note that the POP server checks the SMTP configuration database to ensure that it has been configured with the qualifier /OPTION=TOP_HEADERS so that headers print at the top of the message. If the POP logical name TCPIP$POP_IGNORE_MAIL11_HEADERS is defined, the SMTP option TOP_HEADERS must also be set. If not, the POP server issues a warning in the log file and does not acknowledge the TCPIP$POP_IGNORE_MAIL11_HEADERS definition.

18.1.6.1 How POP Rebuilds the OpenVMS Mail From: Field

The most important message header is the `From:` header, because it can be used as a destination address if a reply is requested from the POP client. Therefore, the POP server rebuilds the OpenVMS Mail `From:` field in compliance with RFC 822 before sending the header to the POP client.

The different types of addresses that can appear in the OpenVMS Mail `From:` field are as follows:

Address Type	Address Format
SMTP	SMTP%"*legal-address*," where *legal-address* is an address that is compliant with RFC 822 and is commonly in the *user@domain* format
DECnet	*node::username*
User name	*username*
DECnet address within quotation marks	*node::"user@host"*
Cluster-forwarding SMTP address	*node*::SMTP%*"user@domain"*

A host name is local if one of the following is true:

- The host name is the same as the substitute domain specified in the SMTP configuration.
- The host name is found in the TCPIP$SMTP_LOCAL_ALIASES.TXT file.

Some POP client systems are confused by the use of personal names when you attempt to reply to a mail message or when the name contains commas or other special characters. If you define the TCPIP$POP_PERSONAL_NAME logical name outlined in Section 18.3, make sure you test the configuration carefully with your POP client systems.

The following sections describe how POP rebuilds the message `From:` field for each type of address.

18.1.6.1.1 SMTP Address The POP server uses the SMTP address within the quotation marks to rebuild the `From:` field of an SMTP address. For example, message header `From: SMTP%"james.jones@federation.gov"` becomes:

```
From: james.jones@federation.gov
```

SMTP hides nested quotation marks by changing them to cent sign (¢) characters before passing them to OpenVMS Mail and then changing them back after a reply. The POP server removes any cent signs that designate double quotation marks. For example, the following message header:

```
From: SMTP%"¢ABCMTS::MRGATE::\¢ABCDEF::VIVALDI \¢¢@xyz.org"
```

Becomes:

```
From: "ABCMTS::MRGATE::\"ABCDEF::VIVALDI\""@xyz.org"
```

18.1.6.1.2 DECnet Address The TCPIP$POP_DECNET_REWRITE logical name values define how the POP server rebuilds a DECnet address, as shown in the following list:

- GENERIC

  The entire address is changed to the SMTP format. For example, from host `widgets.xyzcorp.com`, the message header `From: ORDERS::J_SMITH` becomes:

  ```
  From: "ORDERS::J_SMITH"@widgets.xyzcorp.com
  ```

- NONE

  The `From:` line is sent to the POP client unmodified. For example:

  ```
  From: ORDERS::J_SMITH
  ```

  You cannot reply to this type of message because the SMTP server does not accept an address in this form.

- TRANSFORM

  The POP server attempts to translate the DECnet node name to a TCP/IP host name. If the name can be translated, the POP server checks to see whether the translated host name is local. If so, the `From:` header becomes an address in the form *user@substitute-domain*. If not, the `From:` header becomes an address in the form *user@hostname*. Note that the POP and SMTP servers call the same routine to determine if a host name is local.

  The following examples show some ways the POP server translates DECnet node names to TCP/IP node names. In these examples:

  - The local host name is `orders.acme.widgets.com`
  - ORDERS translates to `"orders.acme.widgets.com"`
    * The message header `From: ORDERS::J_SMITH` becomes:

      ```
      From: j_smith@orders.acme.widgets.com
      ```

    * For a substitute domain of `acme.widgets.com`, the message header `From: ORDERS::J_SMITH` becomes:

      ```
      From: j_smith@acme.widgets.com
      ```

    * If HOST12 translates to `host12.acme.widgets.com`, which is not local on `orders.acme.widgets.com`, the message header `From: HOST12::J_JONES` becomes:

      ```
      From: j_jones@host12.acme.widgets.com
      ```

    * If HOST13 does not translate and host `orders.acme.widgets.com` has no substitute domain defined, the message header `From: HOST13::J_JONES` becomes:

      ```
      From: "HOST13::J_JONES"@orders.acme.widgets.com
      ```

18.1.6.1.3 User Name-Only Address If an SMTP substitute domain is defined, the POP server appends it to the user name, followed by a commercial at sign (@). Otherwise, POP uses the local host name.

For example, with a substitute domain defined as `acme.widgets.com`, the message header `From: Smith` becomes:

```
From: smith@acme.widgets.com
```

18.1.6.1.4 DECnet Address That Contains Quotation Marks The values assigned to the TCPIP$POP_QUOTED_DECNET_REWRITE logical name define how the POP server rebuilds a DECnet address that contains quotation marks. The values are:

- GENERIC

  The address is changed to the SMTP format. For example, on host `widgets.xyzcorp.com`, the message header `From: ORDERS::"j_smith@acme.com"` becomes:

  ```
  From: "ORDER::\"j_smith@acme.com\""@widgets.xyzcorp.com
  ```

- NONE

  The `From:` line is passed to the POP client without being modified. For example:

  ```
  From: ORDERS::"j_smith@acme.com"
  ```

You cannot reply to this type of mail message because the SMTP server does not accept an address of this form.

- TRANSFORM

  The POP server uses the text inside the quotation marks. For example, the message header `From: ORDERS::"j.smith@acme.com"` becomes:

  ```
  From: j.smith@acme.com
  ```

18.1.6.1.5 Cluster-Forwarding SMTP Address With a cluster-forwarding SMTP address, the POP server uses the SMTP address within the quotation marks. For example, the message header `From: ABCDEF::SMTP%"james.jones@federation.gov"` becomes:

```
From: james.jones@federation.gov
```

18.1.6.1.6 All Other Addresses For all other address formats, the POP server changes the entire address to the SMTP format:

- Quotation marks in the address are prefixed with the backslash (\) escape character.
- The entire address is placed within quotation marks.
- A commercial at sign (@) is appended.
- If the SMTP substitute domain is configured, it is appended. Otherwise, the name of the local host is appended.

For example, if the substitute domain is `xyz.org`, the message header `From: ABCMTS::MRGATE::"ORDERS::SPECIAL"` becomes:

```
From: "ABCMTS::MRGATE::\"ORDERS::SPECIAL\""@xyz.org
```

If the logical name TCPIP$POP_IGNORE_MAIL11_HEADERS is defined and the address is an SMTP address, the rebuilt `From:` field is not displayed to the user. In this case, the POP server sends the actual headers from the body of the mail as the mail headers.

18.2 POP Server Startup and Shutdown

The POP server process starts automatically if you specified automatic startup during the configuration procedure (TCPIP$CONFIG.COM).

The POP server can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$POP_STARTUP.COM allows you to start up the POP server independently.
- SYS$STARTUP:TCPIP$POP_SHUTDOWN.COM allows you to shut down the POP server independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$POP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the POP server is started.

- SYS$STARTUP:TCPIP$POP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the POP server is shut down.

18.3 Modifying POP Server Characteristics

To modify the default POP server settings and configure additional characteristics, define TCPIP$POP logical names in the POP_SYSTARTUP.COM file. If you modify the POP startup file, restart the POP server to make the changes take effect.

You can modify the following POP server characteristics:

- Security levels
- Error-message logging
- Maximum number of mail messages downloaded per connection
- Link idle time
- Mail header options
- Ability to set the size of the TCP flow control buffer
- Ability to disable the USER and PASS commands
- Ability to purge mail messages

Table 18–2 outlines the POP logical names, default settings, and characteristic options.

Table 18–2 POP Logical Names

Logical Name	Description

TCPIP$POP_SECURITY *value*	Defines a level of security for the POP server. Determines the timing and text of error messages sent from the POP server to the POP client when authorization errors occur (for example, when an invalid user name or password is sent):
	• FRIENDLY (default)
	The error messages provide information about a particular error. For example, if a password is incorrect, the client receives the following error message:
	`-ERR password supplied for "jones" is incorrect`
	• SECURE
	One error message is sent in response to all authorization errors except when an invalid user name is specified. For example:
	`Access to user account "jones" denied`
	When the POP server receives an invalid user name, it replies to the POP client with a `+OK` message. After the POP client sends the password, the POP server sends the `-ERR access denied` message. This method prevents an unauthorized user from knowing whether the access was denied because of an incorrect user name or password.

TCPIP$POP_TRACE	If defined, the POP server records all messages sent to and received from the POP client in a log file.
TCPIP$POP_LOG_LEVEL *value*	Defines the type of messages logged by the POP server: • ERROR Logs only error messages. • INFORMATIONAL (default) Logs informational messages and error messages. • THREAD Logs information about client and server interactions as well as informational and error messages. • DEBUG Logs full diagnostic information. This is used for problem diagnosis.
TCPIP$POP_POSTMASTER *value*	Defines a person or persons to receive a failure mail message from the POP server startup procedure (TCPIP$POP_STARTUP.COM) when the POP server exits with an error. For example, to have the failure mail message sent to users JONES and SMITH, define the logical name as follows: `$ DEFINE/SYSTEM TCPIP$POP_POSTMASTER "JONES, SMITH"`

(continued on next page)

Table 18–2 (Cont.) POP Logical Names

Logical Name	Description
TCPIP$POP_MESSAGE_MAXIMUM *n*	Defines the maximum number of mail messages that a single client can download per connection, where *n* is a number from 0 to 65,535. If not defined, the POP server uses the default value of 0 (no maximum).
TCPIP$POP_LINK_IDLE_TIMEOUT *n*	Determines the length of time the server allows a link to a POP client to remain idle, where *n* is a number specified in OpenVMS delta time delimited by quotation marks. A POP link remains active until it is released by the POP client. If not defined, the POP server does not set a link idle value (0 00:00:00.00).
TCPIP$POP_PERSONAL_NAME	If defined, the POP server provides the POP clients with the message header `From:` fields that include the sender's personal name, if one appeared in the sender's `From:` field.
TCPIP$POP_LEAVE_IN_NEWMAIL	If defined, mail that has been read by the PC client but not deleted remains in the NEWMAIL folder. Allows users to access mail from different systems and determine when to move or delete the mail from the POP server. If not defined, mail that has been read but not deleted is moved to the MAIL folder.
TCPIP$POP_USE_MAIL_FOLDER	If defined, moves all mail to the MAIL folder and displays this folder instead of the NEWMAIL folder.
TCPIP$POP_FAST_SCAN	If defined, the POP server estimates the number of bytes for the size of the mail message based on the number of lines in the message instead of counting the exact number of bytes. Setting this logical may improve performance.
TCPIP$POP_MAXIMUM_THREADS	Allows you to define the number of process threads that POP can activate. The default is 15. If you set this logical to 1, the POP server becomes single threaded. This logical is recommended only as a temporary solution to system resource problems.
TCPIP$POP_IGNORE_MAIL11_HEADERS	If defined, the POP server ignores the OpenVMS message headers when mail is sent from SMTP, which contains an SMTP address in the `From:` field. For information about how POP forms message headers, see Section 18.1.6.
TCPIP$POP_SEND_ID_HEADERS	If defined, the POP server sends `X-POP3-Server` and `X-POP3-ID` headers for each mail message. If not defined, the ID headers are not sent for any mail from an SMTP address. For information about how POP handles message headers, see Section 18.1.6.

(continued on next page)

Table 18–2 (Cont.) POP Logical Names

Logical Name	Description
TCPIP$POP_DECNET_REWRITE *value*	Determines how the POP server rebuilds a simple DECnet address (of the form *node::user*) in the OpenVMS message `From:` field when it sends the mail to the POP client; *value* is one of the following: • GENERIC Simple DECnet addresses are changed to the SMTP address format. • NONE Simple DECnet addresses are sent unmodified to the POP client. • TRANSFORM (default) The POP server attempts to transform the DECnet address into an SMTP address by translating the DECnet node name to a TCP/IP host name. For more information about how POP rebuilds the message headers, see Section 18.1.6.1.2.
TCPIP$POP_QUOTED_DECNET_REWRITE *value*	Determines how the POP server rebuilds a DECnet address that contains quotation marks (an address of the form *node::"user@host"*) in the OpenVMS Mail `From:` field when it sends the message to the POP client; *value* is one of the following: • GENERIC DECnet addresses that contain quotation marks are changed to the SMTP address format. • NONE DECnet addresses that contain quotation marks are sent unmodified to the POP client. • TRANSFORM (default) The POP server uses the text within the quotation marks in the `From:` field it sends to the POP server. For more information about how POP rebuilds the message headers, see Section 18.1.6.1.4.
TCPIP$POP_SNDBUF *n*	Allows you to increase or decrease the size of the TCP flow control buffer. Sets the SO_SNDBUF socket option to a specific number; *n* is the number 512 or greater. If not defined, the POP server uses the value specified in the SHOW PROTOCOL/PARAMETERS command.
TCPIP$POP_DISUSERPASS	Disables the client USER and PASS commands and sends a failure message to the POP client on receipt of either command. For more information about POP user authorization methods, see Section 18.1.5.
TCPIP$POP_PURGE_RECLAIM	If defined, the POP server performs a PURGE/RECLAIM command action after it deletes messages.

18.4 Enabling MIME Mail

The MIME (Multipurpose Internet Mail Extensions) specification provides a set of additional headers you can use so users can send mail messages composed of more than simple ASCII text. MIME is an enhancement to RFC 822.

For MIME mail to be decoded correctly, follow these guidelines:

- Configure the SMTP server with the /OPTION=TOP_HEADERS qualifier, because the first lines of mail text after the four OpenVMS message header lines and the initial separating line must be the MIME headers.
- Configure the POP server with the TCPIP$POP_IGNORE_MAIL11_ HEADERS logical name. Otherwise, MIME headers are not parsed as message headers.
- The OpenVMS message `From:` field must be recognized as an SMTP address. Otherwise, the POP server sends the headers it creates from OpenVMS message headers as the headers of the mail message. For information about POP message headers, see Section 18.1.6.

  Define the logical name TCPIP$SMTP_JACKET_LOCAL to 1 for all SMTP cluster systems, which ensures that the mail will be delivered if the domain in the `From:` or `To:` fields appears local. For example:

  ```
  $ DEFINE/SYSTEM TCPIP$SMTP_JACKET_LOCAL 1
  ```

If MIME mail does not decode, check the mail headers on the client system. If you see multiple blocks of headers and the MIME version header is not in the first block, confirm that you have followed these guidelines.

18.5 Solving POP Problems

The following sections describe ways to troubleshoot problems associated with using the POP server. Some of these include:

- Reviewing error and OPCOM messages sent to the log file
- Simulating a POP client and entering XTND commands

18.5.1 POP Server Messages

Many of the problems encountered using POP pertain to failed or misinterpreted commands or authorization errors. As the first step toward solving problems, you should review the messages provided by the POP server.

The POP server logs command error and OPCOM (authorization) messages in the file SYS$SYSDEVICE:[TCPIP$POP]POP_RUN.LOG. By default, the POP server sends informative error messages to the client about specific errors.

If the SERVICE database log option REJECT is set, the POP server sends OPCOM messages when it rejects POP client commands because of authorization failures. These errors include the receipt of a client's USER command with an invalid user name, or a PASS command with an invalid password.

By default, OPCOM messages are displayed on the client system and are listed in the log file. To disable OPCOM messages, disable the REJECT logging option for the POP service, as follows:

```
$ TCPIP SET SERVICE POP/LOG=NOREJECT
```

18.5.2 Using POP Extension Commands

For troubleshooting purposes, you can simulate a POP client and enter the XTND commands listed in Table 18–3 to obtain information.

Table 18–3 POP Extension (XTND) Commands

Command	Action
XTND CLIENT	Logs POP client information (if the client supplies it). Helpful for troubleshooting if you use POP with a variety of POP clients that identify themselves.
XTND LOGLEVEL	Dynamically adjusts POP logging level. Supported levels are INFORMATIONAL (default), ERROR, THREAD, and DEBUG.
XTND STATS	Displays POP statistics in the following format: +OK Statistics follow Version Number : TCPIP X5.0, OpenVMS V7.1 Alpha Logging Level : DEBUG Current Time : 1999-04-06 06:13:46 Start Time : 1999-04-04 06:42:17 CPU Seconds : 7.89 (0 mins, 7 secs) Current Threads : 1 Total Threads : 6 Max Threads : 1 Too Many Threads : 0 Normal Disconnects : 5 Abnormal Disconnects : 0 Client Timeouts : 0 Blocked Socket Count : 0 Retrieved Messages : 4 Retrieved Octets : 1102 Average Octets : 275 Minimum Octets : 222 Maximum Octets : 319 Auth Failures : 1 Current Users : 0. smith
XTND SHUTDOWN	Performs an orderly shutdown of POP. Waits for current client connections to disconnect. Recommended over the DCL command STOP.

To simulate a POP client and obtain information:

1. Enter the TELNET command to the POP port (110).
2. Using the USER and PASS command, enter your user name and password.
3. Enter an XTND command.

For example:

```
$ TELNET UCXSYS 110

%TELNET-I-TRYING, Trying ... 16.20.208.53
%TELNET-I-SESSION, Session 01, host ucxsys, port 110
+OK  POP server TCPIP  Version 5.0, OpenVMS V7.1 Alpha at ucxsys.acme.com, up
since 1999-04-04 06:42:17 <24A00E61._6_APR_1999_06_02_31_15@ucxsys.acme.com>

USER username

 +OK Password required for "username"

PASS password

 +OK Username/password combination ok
```

```
XTND LOGLEVEL DEBUG
 +OK logging level changed to debug
QUIT
 +OK TCPIP  POP server at ucxsys.acme.com signing off.
```

19

Configuring XDMCP-Compatible X Displays

The X Window System, developed by the Massachusetts Institute of Technology, is a network-based graphics window system based on the client/server application model. The X protocol, through which the client and server communicate, runs on UNIX domain sockets, TCP/IP, or DECnet. This means that an X display on one system can display information output from an application running on another system in the network.

An X display is a graphic output device that is known by The X Display Manager (XDM), such as:

- An X terminal
- A workstation that has the X Window System software installed and configured
- A PC running Windows or Windows NT and some X Window System software, such as eXcursion or Exceed

This chapter reviews key concepts, discusses how to configure an XDMCP-compatible X display using the TCP/IP Services XDM server, and covers the following topics:

- XDMCP queries (Section 19.2)
- XDM configuration files (Section 19.3)
- XDM log files(Section 19.4)
- XDM server startup and shutdown (Section 19.5)
- Configuring the XDM server(Section 19.6)
- Configuring other X displays (Section 19.7)

19.1 Key Concepts

The X Display Manager (XDM) is an X client that manages the login process of a user's X window session. XDM is responsible for displaying a login screen on a display specified by an X server, establishing an X window session, and running scripts that start other X clients. When the user logs out of the X session, XDM is responsible for closing all connections and resetting the terminal for the next user session.

An earlier version of XDM had limitations that were resolved with the introduction of the XDM Control Protocol (XDMCP). Before XDMCP, XDM used the XSERVERS file to keep track of the X terminals for which it managed the login process. At startup, XDM initialized all X terminals listed in the XSERVERS file. If the X terminal was turned off and then on again, XDM had no way of knowing that a new login process should be initiated at the X terminal.

To reinitialize the X terminal, the XDM process had to be restarted. This problem was solved through the development of the XDM Control Protocol.

Now, because of XDMCP, XDM can listen for management requests from X terminals as well as use the XSERVERS file for the X terminals that were not XDMCP compatible. Most X terminals today are XDMCP compatible.

The TCP/IP Services implementation of XDM is based on the X11R6.1 release from X Consortium.

19.2 XDMCP Queries

XDMCP provides the following methods to query XDM for service:

- Direct

  X terminals, configured for the direct request method, send a connection request to a specific host.

- Broadcast

  X terminals, configured for a broadcast request, send out a general query to ask for service from all nodes running XDM. A list of the responding nodes is then presented to the user for selection by the client software.

- Indirect

  An indirect request is used to relay a request for service from one XDM node to another.

The TCP/IP Services implementation of XDM does not support the indirect query with a chooser box supported by some other XDM servers.

An authentication protocol is supported for all three types of requests.

19.3 XDM Configuration Files

If the files are present, XDM uses the following files to configure the X display environment:

- Master configuration (XDM_CONFIG.CONF)
- Servers (XSERVERS.TXT)
- Access (XACCESS.TXT)
- Keys (XDM_KEYS.TXT)
- Session (XDM_XSESSION.COM)

After installing XDM, you can use the TCP/IP Services-supplied configuration templates located in SYS$SPECIFIC:[TCPIP$XDM] to create the configuration files. The default directory location of the configuration and template files is the SYS$SPECIFIC:[TCPIP$XDM] directory.

19.3.1 Master Configuration File

The master configuration file, which is an optional file, specifies the location and file names of the other configuration files used to control the operation of XDM.

Example 19–1 shows the contents of the default configuration template file (XDM_CONFIG.TEMPLATE) supplied with XDM:

Example 19–1 XDM_CONFIG.TEMPLATE File

```
!
! Default SYS$SPECIFIC:[TCPIP$XDM]XDM_CONFIG.CONF file
!
DisplayManager.keyFile:            SYS$SPECIFIC:[TCPIP$XDM]XDM_KEYS.TXT
DisplayManager.servers:            SYS$SPECIFIC:[TCPIP$XDM]XSERVERS.TXT
DisplayManager.accessFile:         SYS$SPECIFIC:[TCPIP$XDM]XACCESS.TXT
DisplayManager*RemoveDomainname:   true
```

Each noncomment line in the file must consist of a keyword and value pair. TCP/IP Services supports the following keywords:

```
$ TYPE SYS$SPECIFIC:[TCPIP$XDM]XDM_CONFIG.TEMPLATE
!
! Default SYS$SPECIFIC:[TCPIP$XDM]XDM_CONFIG.CONF file
!
DisplayManager.keyFile:            SYS$SPECIFIC:[TCPIP$XDM]XDM_KEYS.TXT
DisplayManager.servers:            SYS$SPECIFIC:[TCPIP$XDM]XSERVERS.TXT
DisplayManager.accessFile:         SYS$SPECIFIC:[TCPIP$XDM]XACCESS.TXT
DisplayManager*RemoveDomainname:   true
```

The file specification for the master configuration file is:

```
SYS$SPECIFIC:[TCPIP$XDM]XDM_CONFIG.CONF
```

XDM uses the `DisplayManager*RemoveDomainname:` value when computing the display name for XDMCP clients. BIND, when performing a host name lookup creates a fully qualified host name for the X terminal. When this keyword is set to TRUE, XDM removes the domain name portion of the host name if it is the same as the local host domain. The default value of `DisplayManager*RemoveDomainname:` is TRUE.

19.3.2 XACCESS.TXT File

The XACCESS.TXT file, a required file, allows or restricts access to remote X servers. If the XACCESS.TXT file is not present, the system restricts all remote X server access. You use this file to control the way XDM responds to broadcast, direct, and indirect requests from X servers.

The default file specification for the XACCESS.TXT configuration file is:

```
SYS$SPECIFIC:[TCPIP$XDM]XACCESS.TXT
```

If you choose to use another file name or directory location, you can override the default by adding a line in the XDM_CONFIG.CONF file similar to the following:

```
DisplayManager.accessFile:  WORK1$:[XDM]XACCESS.TXT
```

Example 19–2 shows a sample XACCESS.TXT configuration file:

Example 19–2 XACCESS.TXT File

```
# $XConsortium: Xaccess,v 1.5 91/08/26 11:52:51 rws Exp $
#
# Access control file for XDMCP connections
#
# To control Direct and Broadcast access:
#
#       pattern
#
# To control Indirect queries:
#
#       pattern         list of hostnames and/or macros ...
#
# To define macros:
#
#       %name           list of hosts ...
#
# The first form tells xdm which displays to respond to itself.
# The second form tells xdm to forward indirect queries from hosts matching
# the specified pattern to the indicated list of hosts.
#
# In all cases, xdm uses the first entry which matches the terminal;
# for IndirectQuery messages only entries with right hand sides can
# match, for Direct and Broadcast Query messages, only entries without
# right hand sides can match.
#

*               #any host can get a login window

#
# To hardwire a specific terminal to a specific host, you can
# leave the terminal sending indirect queries to this host, and
# use an entry of the form:
#

#terminal-a     host-a
```

Allowing Direct Access

To allow access, add a line to the XACCESS.TXT file with the host name, as shown in the following example:

```
condor.compaq.com
```

Denying Access

To restrict access, add a line to the XACCESS.TXT file with the host name, preceded by an exclamation point, as shown in the following example:

```
!rufus.compaq.com
```

You can use the question mark (?) and the asterisk (*) wildcard characters to specify host names that vary with one character or more than one character.

Allowing Indirect Access

To allow indirect access, add a line to the XACCESS.TXT file similar to the following line:

```
rufus.compaq.com     richard.compaq.com henry.compaq.com william.compaq.com
```

19.3.3 XSERVERS.TXT File

The XSERVERS.TXT file was originally used to specify all X servers to be managed by XDM. However, since the introduction of XDMCP, there is no need to specify X servers that are XDMCP compatible in this file.

This file now specifies the X servers that do not support XDMCP. Unlike other XDM implementations, this file is not used to specify XDM support for the local display server.

The default file specification for the XSERVERS.TXT file is:

```
SYS$SPECIFIC:[TCPIP$XDM]XSERVERS.TXT
```

If you choose to use a different name and directory location, you can override the default by adding a line to the XDM_CONFIG.CONF file similar to the following line:

```
DisplayManager.servers:  WORK1$:[XDM]XSERVERS.TXT
```

Example 19–3 shows a sample XSERVERS.TXT configuration file:

Example 19–3 XSERVERS.TXT File

```
$ TYPE XSERVERS.TXT
#
#
# This file can be used to support X terminals which do not support XDMCP.
#
# For each terminal, add a line that consists of
#     DisplayName:0 foreign
#
# Where DisplayName is a IP name.
#
rufus.compaq.com:0 foreign
```

The word foreign in the previous example indicates that the X server is running on another machine.

19.3.4 XDM_KEYS.TXT File

The XDM_KEYS.TXT file provides XDM-AUTHENTICATION-1 style XDMCP authentication. This optional file contains key ID and key value pairs for use with X terminals that support or require XDM authorization.

Each noncomment line in the XDM_KEYS.TXT file contains a display ID and a key value. The file is used when a request containing a display ID key is received from an X terminal. The corresponding key value is encrypted and returned to the X terminal. If the key value in the configuration file matches the key value specified by the X terminal's control information, the session is allowed.

The default file specification for the XDM_KEYS.TXT files is:

```
SYS$SPECIFIC:[TCPIP$XDM]XDM_KEYS.TXT
```

If you choose to use a different name and directory location, you can override the default by adding a line to the XDM_CONFIG.CONF file similar to the following line:

```
DisplayManager.keyFile:  WORK1$:[XDM]XDM_KEYS.TXT
```

Example 19–4 shows a sample XDM_KEYS.TXT configuration file:

Example 19–4 XDM_KEYS.TXT

```
#
# Security Key File
#
#  Excursion Display ID                Excursion Cookie:
#
test123456                            123457
#
#  Exceed Display ID:                  Exceed Key:
#
HCLpcXserver:629409365                1234568
#
```

19.3.5 XDM_XSESSION.COM File

The XDM_XSESSION.COM file is an optional command procedure file that specifies the type of X window that XDM displays after a user has successfully logged in.

XDM's default operation is to create a Common Desktop Environment (CDE) using the commands from the SYS$SYSTEM:TCPIP$XDM_XSESSION.COM file:

```
$ DEFINE DECW$DISPLAY "''p1'"
$ DEFINE display      "''p1'"
$ @CDE$PATH:XSESSION.COM
```

At present, CDE is only available on Alpha systems in version 1.2-4 or later of DWMOTIF, and not at all on VAX systems. If the CDE command procedure XSESSION.COM is not found on the system, XDM will look for the DECwindows Desktop Session Manager startup command procedure, DECW$STARTSM.COM to initiate the session using the commands:

```
$ SET DISPLAY/CREATE/NODE=nodename/TRANSPORT=TCPIP
$ @SYS$MANAGER:DECW$STARTSM.COM
```

Before executing either of these command procedures, XDM looks for an XDM_XSESSION.COM file in the user's SYS$LOGIN directory. If found, XDM executes the file. Users can create a DECterm by adding the following DCL commands to their XDM_XSESSION.COM file:

```
$ SET PROC/PRIV=SYSNAM
$ SET DISPLAY/CREATE/NODE=workstation_display/TRANSPORT=TCPIP -
_$ /EXECUTIVE_MODE
$ CREATE/TERMINAL/WAIT/WINDOW_ATTRIBUTES=(ICON=nodename, -
_$  TITLE=window_title)
```

For a complete description of the CREATE and SET DISPLAY commands and their qualifiers, use the DCL command HELP at the OpenVMS system prompt.

19.4 XDM Log Files

XDM maintains three log files to record XDM server and client activity:

- XDM server log file
- X terminal process log file
- User process log file

Table 19–1 lists the XDM log files and their OpenVMS directory locations.

Table 19–1 XDM Log Files

Process	File Name	Location
XDM server	TCPIP$XDM_RUN.LOG	SYS$SPECIFIC:[TCPIP$XDM]
X terminal	*xterm_name_domain*.COM	SYS$SPECIFIC:[TCPIP$XDM.WORK]
	xterm_name_domain.ERR	SYS$SPECIFIC:[TCPIP$XDM.WORK]
	xterm_name_domain.OUT	SYS$SPECIFIC:[TCPIP$XDM.WORK]
User	*xterm_name_domain*.LOG	SYS$LOGIN

19.5 XDM Server Startup and Shutdown

The XDM server can be shut down and started independently from the rest of the TCP/IP Services software. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$XDM_STARTUP.COM allows you to start up the XDM service.
- SYS$STARTUP:TCPIP$XDM_SHUTDOWN.COM allows you to shut down the XDM service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$XDM_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when XDM is started.
- SYS$STARTUP:TCPIP$XDM_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when XDM is shut down.

19.6 Configuring the XDM Server

To configure your XDM server, you need to:

- Run SYS$MANAGER:TCPIP$CONFIG to create the default directories and a user ID for the XDM component. The configuration procedure checks to see whether the following DECwindows components are installed:
  - SYS$COMMON:[SYSLIB]DECW$XLIBSHR.EXE
  - SYS$COMMON:[SYSLIB]DECW$XTLIBSHRR5.EXE
  - SYS$COMMON:[SYSLIB]DECW$TRANSPORT_COMMON.EXE (VAX only)

If the DECwindows components are not found, TCPIP$CONFIG notifies you and gives you the option of configuring XDM, with the assumption that before you attempt to activate XDM you will install the DECwindows components. TCPIP$CONFIG notifies you of this situation with the following prompt:

```
XDM requires DECwindows components that are not installed.
Attempts to activate XDM will fail.

Type C to continue with XDM configuration, or E to exit [ E ]:
```

- If necessary, use the template files located in SYS$SPECIFIC:[TCPIP$XDM] to create an XDM_CONFIG.CONF, XSERVERS.TXT, XACCESS.TXT, and XDM_KEYS.TXT file. These files are not required unless you want to:
  - Configure a non-XDMCP X display
  - Restrict access to a remote X server
  - Provide XDMCP authentication
  - Change the way XDM computes the display name for XDMCP clients.
- Run SYS$MANAGER:TCPIP$CONFIG to enable XDM.

19.6.1 Ensuring XDM Is Enabled and Running

To make sure that the XDM service is enabled and that the XDM process is running, enter the following TCPIP command:

```
$ TCPIP SHOW SERVICE XDM
Service             Port  Proto    Process          Address            State
XDM                  177  UDP      TCPIP$XDM        0.0.0.0             Enabled
```

19.7 Configuring Other X Displays

If you have an X terminal that does not support the XDMCP protocol, you can manage this terminal by using an XSERVERS.TXT configuration file. See Section 19.3.3 for information about how to create the configuration file.

If you are running Compaq eXcursion, refer to the *Compaq PATHWORKS 32 eXcursion User's Guide* for configuration information. For all other X servers, refer to the third-party X Window System software documentation for information about how to configure their product.

Part 5

Network File Services

Part 5 describes how to configure, use, and manage the components that enable transparent network file sharing: NFS server, PC-NFS, and NFS client. It includes the following chapters:

- Chapter 20, NFS Server, describes how to set up the NFS server and make file systems available to users on NFS client hosts. This chapter also describes how to set up PC-NFS, how to troubleshoot server and file system problems, and describes the NFS characteristics that can affect system performance.
- Chapter 21, NFS Client, describes how to set up the NFS client, which provides users with access to remote file systems.

20
NFS Server

The Network File System (NFS) server software lets you set up file systems on your OpenVMS host for export to users on remote NFS client hosts. These files and directories appear to the remote user to be on the remote host even though they physically reside on the local system.

After the NFS server is installed on your computer, you must configure the server to allow network file access.

This chapter reviews key NFS concepts and describes:

- How to start up and shut down the NFS server (Section 20.2)
- How to set up the NFS server in an OpenVMS cluster (Section 20.3)
- How to set up PC-NFS (Section 20.4)
- How to manage the MOUNT service (Section 20.5)
- How to register users and hosts (Section 20.6)
- How to back up the file system (Section 20.7)
- How to set up and export an OpenVMS file system (Section 20.8)
- How to set up and export a container file system (Section 20.9)
- How to manage a container file system (Section 20.10)
- How to set up and manage NFS security controls (Section 20.11)
- How to modify NFS server characteristics (Section 20.12)
- How to modify file system characteristics (Section 20.13)
- NFS file locking (Section 20.14)
- How to improve the performance of NFS operations (Section 20.15)

See Chapter 21 for information on managing the NFS client.

If your network includes PC clients, you may want to configure PC-NFS. Section 20.1.9 and Section 20.4 provide more information.

20.1 Key Concepts

NFS software was originally developed on and used for UNIX machines. For this reason, NFS implementations use UNIX style conventions and characteristics. The rules and conventions that apply to UNIX files, file types, file names, file ownership, and user identification also apply to NFS.

Because the TCP/IP Services product runs on OpenVMS, the NFS software must accommodate the differences between UNIX and OpenVMS file systems, for example, by converting file names and mapping file ownership information. You must understand these differences to configure NFS properly on your system,

to select the correct file system for the application, and to ensure that your file systems are adequately protected while granting access to users on remote hosts.

The following sections serve as a review only. If you are not familiar with NFS, see the *DIGITAL TCP/IP Services for OpenVMS Concepts and Planning* manual for more information.

20.1.1 Clients and Servers

NFS is a client/server environment that allows computers to share disk space and allows users to work with their files from multiple computers without copying them to their local system. The NFS server can make any of its file systems available to the network by **exporting** the files and directories. Users on authorized client hosts access the files by **mounting** the exported files and directories. The NFS client systems accessing your server may be running UNIX, OpenVMS, or other operating systems.

The NFS client identifies each file system by the name of its **mount point** on the server. The mount point is the name of the device or directory at the top of the file system hierarchy that you create on the server. An NFS device is always named DNFS*n*. The NFS client makes file operation requests by contacting your NFS server. The server then performs the requested operation.

20.1.2 NFS File Systems on OpenVMS

The OpenVMS system includes a hierarchy of devices, directories and files stored on a Files–11 On-Disk Structure (ODS-2) formatted disk. OpenVMS and ODS-2 define a set of rules that govern files within the OpenVMS file system. These rules define the way that files are named and catalogued within directories.

If you are not familiar with OpenVMS file systems, refer to the *OpenVMS System Manager's Manual: Essentials* to learn how to set up and initialize a Files–11 disk.

You can set up and export two different kinds of file systems: a traditional OpenVMS file system or a UNIX style file system built on top of an OpenVMS file system. This UNIX style file system is called a **container file system**.

Each file system is a multilevel directory hierarchy: on OpenVMS systems, the top level of the directory structure is the master file directory (MFD). The MFD is always named [000000] and contains all the top-level directories and reserved system files. On UNIX systems or with a container file system, the top-level directory is called the **root**.

20.1.2.1 Selecting a File System

You can set up and export either an OpenVMS file system or a container file system. Which one you choose depends on your environment and the user needs on the NFS client host.

You might use an OpenVMS file system if:

- Your environment calls for extensive file sharing between your OpenVMS system and another OpenVMS host, or between your system and a UNIX client.
- Users on the client need to maintain multiple versions of files.

Select the OpenVMS file system if you need to share files between users on OpenVMS and users on NFS clients.

You might use a container file system if:

- You do not require extensive file sharing between your OpenVMS system and a UNIX client.
- Client applications require symbolic or hard links or special files.

20.1.2.2 Understanding the Container File System

The NFS software lets you create a logical UNIX style file system on your OpenVMS host that conforms to UNIX file system rules. This means that any UNIX application that accesses this file system continues to work as if it were accessing files on a UNIX host.

An OpenVMS server can support multiple container file systems. Creating a container file system is comparable to initializing a new disk with an OpenVMS volume structure, because it provides the structure that enables users to create files. The file system parameters, directory structure, UNIX style file names, and file attributes are catalogued in a data file called a container file.

The number of UNIX containers you should create depends on how you want to manage your system.

In a container file system, each conventional UNIX file is stored as a separate data file. The container file also stores a representation of the UNIX style directory hierarchy and, for each file name, a pointer to the data file. In addition to its UNIX style name, each file in the container file system has a system-assigned valid Files–11 file name.

An OpenVMS directory exists for each UNIX directory stored in the container. All files catalogued in a UNIX directory are also catalogued in the corresponding OpenVMS directory; however, the UNIX directory hierarchy is not duplicated in the OpenVMS directory hierarchy.

Because each UNIX style file is represented as an OpenVMS data file, OpenVMS utilities such as BACKUP can use standard access methods to access these files.

Note

Except for backing up and restoring files, you should not use DCL commands to manipulate files in a container file system. Instead, use the commands described in Section 20.10.

For more information about backing up and restoring files, see Section 20.7 and Section 20.10.7.

For information about setting up container file systems, see Section 20.9.

20.1.3 How the Server Grants Access to Users and Hosts

The server uses the following database files to grant access to users on client hosts:

- The **export database**, TCPIP$EXPORT.DAT, is a collection of entries used to store information about the file systems you want to make available to users on client hosts.

Each entry specifies a directory on the local system and one or more remote hosts allowed to mount that directory. A user on a client host can mount any directory at or below the export point, as long as OpenVMS allows access to the directory. Exporting specific directories to specific hosts provides more control than exporting the root of a file system (or the MFD in an OpenVMS system) to all hosts.

- The **proxy database**, TCPIP$PROXY.DAT, is a collection of entries used to register the identities of users on client hosts. To access file systems on your local server, remote users must have valid accounts on your OpenVMS host.

  The proxy entries map each user's remote identity to a corresponding identity associated with each user's OpenVMS account. When a user on the client host initiates a file access request, the server checks the proxy database before granting or denying the user access to the file.

These database files are usually created by TCPIP$CONFIG and can be shared by all OpenVMS Cluster nodes running TCP/IP Services. To control access to these database files, set the OpenVMS file protections accordingly. By default, World access is denied.

Section 20.6 describes how to create these database files on your server.

20.1.4 How the Server Maps User Identities

Both OpenVMS and UNIX based systems use identification codes as a general method of resource protection and access control. Just as OpenVMS employs user names and UICs for identification, UNIX identifies users with a user name and a user identifier (UID) and one or more group identifiers (GIDs). Both UIDs and UICs identify a user on a system.

The proxy database contains entries for each user who accesses a file system on your local server. Each entry contains the OpenVMS user name, the UID/GID pair that identifies the user's account on the client system, and the name of the client host. This file is loaded into dynamic memory when the server starts.

When a user on the OpenVMS client host requests access to a file, the client searches its proxy database for an entry that maps the requester's identity to a corresponding UID/GID pair. (Proxy lookup is performed only on OpenVMS servers; UNIX clients already know the user by its UID/GID pair.) If the client finds a match, it sends a message to the server that contains the following:

- Identity of the requester as a UID/GID pair
- Requested NFS operation and any data associated with the operation

The server searches its proxy database for an entry that corresponds to the requester's UID/GID pair. If the UID maps to an OpenVMS account, the server grants access to the file system according to the privileges set for that account.

In the following example, the proxy entry maps a client user with UID=15/GID=15, to the OpenVMS account named ACCOUNT2. Any files owned by user ACCOUNT2 are deemed to be also owned by user UID=15 and GID=15.

```
OpenVMS User_name     Type      User_ID    Group_ID  Host_name
ACCOUNT2              OND       15          15       *
```

After the OpenVMS identity is resolved, the NFS server uses this acquired identity for all data access, as described in Section 20.1.7.

20.1.5 Mapping the Default User

In a trusted environment, you may want the server to grant restricted access even if the incoming UID does not map to an OpenVMS account. This is accomplished by adding a proxy entry for the **default user**. The NFS server defines the default user at startup with the following attributes:

- noproxy_uid
- noproxy_gid

You can initialize these attributes using the SYSCONFIG command, which is defined by the SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM procedure. For example:

```
$ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS
$ SYSCONFIG -r nfs_server noproxy_uid=-2 noproxy_gid=-2
```

If the server finds a proxy entry for the default user, it grants access to OpenVMS files as the OpenVMS user associated with "nobody" in the proxy record. TCP/IP Services normally uses the UNIX user "nobody" (–2/–2) as the default user.

To temporarily modify run-time values for the default user, use the /UID_DEFAULT and /GID_DEFAULT qualifiers to the SET NFS_SERVER command.

To permanently modify these values, edit the SYS$STARTUP:TCPIP$NFS_SYSTARTUP.COM file with the commands to define new values for the UID and GID logical names. See Section 20.12 for instructions on modifying SYSCONFIG variables to change the default values.

If you require tighter restrictions, you can disable the default user mapping and set additional security controls by setting the attribute noproxy_enabled. See Section 20.11 for more information.

Note

The configuration procedure for the NFS client creates a nonprivileged account with the user name TCPIP$NOBODY. You may want to add a proxy record for the default user that maps to the TCPIP$NOBODY account.

20.1.6 Mapping a Remote Superuser

When a remote UNIX client does a mount, it is often performed by the superuser. (In some UNIX implementations, this can be performed only by the superuser.)

A superuser (root) on a remote client does not automatically become a privileged user on the server. Instead, the superuser (UID=0) is mapped to the default user defined with the attributes noproxy_uid and noproxy_gid. (By default, user "nobody" (–2/–2) is used.)

You may have remote clients that use the superuser to mount file systems. If you want to grant normal root permissions, add a proxy record with UID=0/GID=1 and map this to an appropriate OpenVMS account. The ability of the remote superuser to mount and access files on the server is controlled by the privileges you grant for this OpenVMS account.

20.1.7 How OpenVMS and the NFS Server Grant File Access

To protect your exported file systems, you must take care when granting account and system privileges for remote users. You must also understand how OpenVMS grants access to files.

The NFS server uses the proxy database to map the incoming user identity to an OpenVMS account. The server uses the account's UIC to evaluate the protection code, along with other security components, before granting or denying access to files.

When a user tries to access a protected file or directory, the OpenVMS system uses the following sequence to compare the security profile of the user against the security profile of the target file or directory.

1. Evaluates the access control list (ACL).

   ACL protection is an OpenVMS feature that grants or denies access to a file based on a rights identifier. If the object has an ACL, the system scans it, looking for an entry that matches any of the user's rights identifiers. If a matching access control entry (ACE) is found, the system either grants or denies access based on the ACE.

   When setting up the NFS server environment, Compaq recommends that you set ACLs to deny access. This forces OpenVMS protection checking, as described in Section 20.1.8.

2. Evaluates the protection code.

   If the ACL does not grant access, the operating system evaluates the system and owner fields of the protection code and grants or denies access based on the relationship between the UIC and the object's protection code.

   OpenVMS uses user identifier codes (UICs) to control file ownership and protection. A UIC is a 32-bit value that consists of a 14-bit group number and a 16-bit member number. Each user of the system has a UIC defined in the SYSUAF file. Access to objects depends on whether the UIC of the process doing the accessing matches the UIC of the object (the file or directory).

3. Looks for special privilege.

   If access was not granted by the ACL or the protection code, the operating system evaluates privileges. Users with system privileges (BYPASS, GROUP, READALL, SYSPRV) may be entitled to access regardless of the protection offered by the ACLs or protection code.

For a more thorough discussion on access checking, refer to the *OpenVMS Guide to System Security*.

20.1.8 Understanding the Client's Role in Granting Access

Before sending a user request to the NFS server, the client performs its own access checks. This check occurs on the client host and causes the client to grant or deny access to data. This means that even though the server may grant access, the client may deny access before the user's request is even sent to the server host. If the client user maps to an OpenVMS account that is not allowed access to a file, an ACL entry may not allow access from an NFS client as it would locally for that OpenVMS account.

It is also possible for the server to reject an operation that was otherwise allowed by the client. With the attribute `noproxy_enabled`, you can use the ACL for additional access control. See Section 20.11 for a complete description of the security features set with this variable.

With this variable set, the TCP/IP Services startup procedure creates the TCPIP$NFS_REMOTE identifier. For example, you can use this identifier in the ACL to reject access to some (or all) files available through NFS. (See Section 20.12 for more information about logical names.)

20.1.9 Granting Access to PC-NFS Clients

TCP/IP Services provides authentication services to PC-NFS clients by means of PC-NFS. As with any NFS client, users must have a valid account on the NFS server host, and user identities must be registered in the proxy database.

Because PC operating systems do not identify users with UID/GID pairs, these pairs must be assigned to users. PC-NFS assigns UID/GID pairs based on information you supply in the proxy database.

The following describes this assignment sequence:

1. The PC client sends a request for its UID/GID pair. This request includes the PC's host name with an encoded representation of the user name and password.
2. PC-NFS responds by searching the proxy database and SYSUAF for a matching entry and by checking the password.

   If a matching entry is located, PC-NFS returns the UID/GID pair to the PC client. The PC stores the UID/GID pair for later NFS requests.
3. If PC-NFS does not find an entry for the PC client in the proxy database, it maps the PC client to the default user TCPIP$NOBODY account. In this case, restricted access is granted based on privileges established for the default user account. See Section 20.1.5 for more discussion on the default user.

20.2 NFS Server Startup and Shutdown

The NFS server can be shut down and started independently. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$NFS_SERVER_STARTUP.COM allows you to start up the NFS server independently.

  When it detects a request from a client host, the auxiliary server starts the NFS server. The NFS server startup command procedure enables the server for automatic startup.
- SYS$STARTUP:TCPIP$NFS_SERVER_SHUTDOWN.COM allows you to shut down the NFS server independently.

  You can stop the NFS server even though clients still have file systems mounted on the server. If a client has a file system mounted with the `hard` option of the UNIX `mount` command, and the client accesses the file system while the server is down, the client will stall while it is waiting for a response from the server.

  Alternatively, if the client has a file system mounted using the `soft` option of the UNIX `mount` command, the client will receive an error message if it attempts to access a file.

Because the NFS protocol is stateless, clients with file systems mounted on the server do not need to remount when the server is restarted. To ensure this uninterrupted service, you must be sure all file systems are mapped before restarting the NFS server. The simplest way to do this is to use the SET CONFIGURATION MAP command.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$NFS_SERVER_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NFS server is started.
- SYS$STARTUP:TCPIP$NFS_SERVER_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NFS server is shut down.

20.3 Running the NFS Server on an OpenVMS Cluster System

If the NFS server resides on more than one host in an OpenVMS Cluster system, you can manage the proxy database and the export database as a homogeneous OpenVMS Cluster system (one proxy file on the OpenVMS Cluster system) or a heterogeneous OpenVMS Cluster system (a different proxy database on each host in the cluster).

The NFS server automatically responds to the requests it receives on any TCP/IP network interface. Therefore, if several OpenVMS Cluster nodes have Internet cluster interfaces, the server can execute as a clusterwide application. Clients that mount file systems using the cluster alias can then be served by any of the NFS servers in the cluster. Because NFS uses cluster failover, if one of the servers is taken down, client requests are redirected to another host in the cluster.

To allow NFS clients to access the cluster, define a cluster alias and a network interface name for each cluster member.

20.4 Setting Up PC-NFS

If you plan to export file systems to PC-NFS client hosts, you must enable PC-NFS using TCPIP$CONFIG. The PC-NFS process starts automatically.

You can also use the following commands to manage PC-NFS:

- DISABLE SERVICE PCNFS (temporarily disables PC-NFS)
- ENABLE SERVICE PCNFS (enables PC-NFS)
- SHOW SERVICE PCNFS (displays information used for troubleshooting)

For information about setting up PC-NFS for printing, see Chapter 24.

20.5 Managing the MOUNT Service

The MOUNT service responds to Version 1 of the MOUNT protocol, which is used with Version 2 of the NFS protocol. It also supports Version 3 of the MOUNT protocol, which is used with Version 3 of the NFS protocol.

The MOUNT service is started automatically when you start the NFS server (for example, using TCPIP$NFS_SERVER_STARTUP.COM).

You can customize the operation of the MOUNT service by using SYSCONFIG to modify the attributes listed in Table 20–1.

Table 20–1 MOUNT Attributes

Attribute	Description
mountd_option_a	Verifies the Internet addresses of hosts that make mount and unmount requests. If a client's address cannot be translated into a host name by the gethostbyaddr( ) function and is then translated back into the same Internet address by the gethost-byname( ) function, the request is rejected. Requires name resolution to be enabled.
mountd_option_d	Turns on Internet address verification and domain checking. If you are running the BIND service, MOUNT verifies that a host making a mount or unmount request is in the server's domain.
mountd_option_i	Verifies the Internet address of hosts that make mount and unmount requests. If a client's address cannot be translated into a host name by the gethostbyaddr( ) function, the request is rejected. Requires name resolution to be enabled. If the mountd_option_i attribute is not set, and a client's address cannot be translated, the address is converted to a string in the form *xx.xx.xx.xx*. This allows users to access exported file systems that have a wildcard (*) (allow everybody) as their host list. The mountd_option_i attribute is automatically enabled when either the mountd_option_d or the mountd_option_s attribute is specified.
mountd_option_n	Allows nonroot mount requests to be served. In previous versions of TCP/IP Services, the servicing of nonroot mount requests was allowed by default. With this version, this attribute must be set to allow nonroot mount requests. Specify this attribute only if there are clients (such as desktop computers) that require it.
mountd_option_s	Turns on Internet address verification and subdomain checking. If you are running the BIND service, the MOUNT service verifies that a host making a mount or unmount request is in the server's domain or subdomain.

See Section 20.12 for information about using the SYSCONFIG command.

20.6 Registering Users and Hosts

In a NFS environment shared by UNIX hosts, a common user authorization domain may be used. In this configuration, each user's UID is unique for all the hosts. On OpenVMS, however, the user authorization file (UAF) cannot be shared, but client user identifiers must be mapped to OpenVMS accounts. Therefore, users on client hosts must have corresponding OpenVMS accounts on the OpenVMS NFS server.

To establish this common allocation on OpenVMS, each client UID must be mapped to a unique OpenVMS account. This arrangement requires a separate OpenVMS account for each NFS client. It is possible to use the same OpenVMS account for multiple users, but this is not recommended for accounts in which users have read or edit access to files.

After setting up appropriate accounts, you must register users in the proxy database and set mount points in the export database.

20.6.1 Adding Proxy Entries

Each user accessing your local server must be registered in the proxy database. See Section 20.1.3 if you are not familar with how the server uses this database to grant access to remote users. You should create the proxy database before the NFS server starts. If you are adding proxies, create the OpenVMS accounts before creating the proxy entries.

An empty proxy database file, TCPIP$PROXY.DAT, is created for you when you first use the TCPIP$CONFIG configuration procedure to configure NFS. This file is empty until you populate it with proxy entries for each NFS user. If you do not use TCPIP$CONFIG to configure NFS, use the CREATE PROXY command to create the empty database file. The file TCPIP$PROXY.DAT resides in the SYS$COMMON:[SYSEXE] directory.

Use the ADD PROXY, REMOVE PROXY, and SHOW PROXY commands to maintain the proxy database. Enter these commands at the TCPIP prompt:

```
TCPIP> ADD PROXY user_name /UID=nn /GID=nn /HOST=host_name
```

For example, you can use the following command to register a user:

```
TCPIP> ADD PROXY SMITH /UID=53 /GID=45 /HOST="june"
```

You can specify a list of hosts for which the UID and GID are valid. For example:

```
TCPIP> ADD PROXY SMITH /UID=53 /GID=45 /HOST=("APRIL","MAY","JUNE")
```

You can also specify that all hosts are valid using an asterisk (*) wildcard character. For example:

```
TCPIP> ADD PROXY SMITH /UID=53 /GID=45 /HOST=*
```

20.6.2 Adding Entries to the Export Database

If you use the configuration procedure to configure NFS, the export database is created for you, if it does not already exist. This file is empty until you populate it with mount point entries. If you do not use TCPIP$CONFIG to configure NFS, use the CREATE EXPORT command to create the empty database file.

Use the ADD EXPORT, REMOVE EXPORT, and SHOW EXPORT commands to maintain the export database. Enter these commands at the TCPIP prompt:

```
TCPIP> ADD EXPORT "/path/name" /HOST=host_name
```

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for more information about these commands and command qualifiers.

You can identify mount points by any of the following methods:

- OpenVMS device name
- A device name and directory
- A logical name

20.7 Backing Up a File System

You can back up NFS-mounted files using standard OpenVMS backup procedures. For more information, see the OpenVMS documentation.

If you back up an OpenVMS file system or a container file system while remote users are accessing the files, the resulting save set may contain files that are in an inconsistent state. For a container file system, there is the additional danger that the container file itself may be in an inconsistent state.

Furthermore, the OpenVMS Backup utility does not issue warning messages when backing up files that are opened by the NFS server, even when the /IGNORE=INTERLOCK qualifier to the BACKUP command was not used.

The approach to backing up is to schedule the backup for a time when users will not be accessing the files. Then either unmap the file systems to be backed up or shut down the NFS server.

If you perform an incremental backup (using the /SINCE=MODIFIED qualifier to the BACKUP command) on container file systems, a separate copy of the container must also be backed up because the container file's modification date never changes. See Section 20.9 for information about setting up container file systems; see Section 20.10 for information about managing container file systems.

20.8 Setting Up and Exporting an OpenVMS File System

The following example describes how to set up an OpenVMS file system on the OpenVMS server and how to make the file system available to Joe Brown, a user on UNIX client `ultra`.

Joe Brown has an OpenVMS user name of BROWN and a UNIX user name of `joe`.

1. Log in to a UNIX node to find the UID/GID for the UNIX user `joe` by entering the following command:

   ```
   % grep joe /etc/passwd
    joe: (encrypted password) :27:58: ...
   ```

   The fields `:27:58` of the password entry for `joe` are the UID and GID. In this example, `joe` has UID=27 and GID=58.

2. Log in to the OpenVMS server.

   The OpenVMS files exist on DSA301:[BROWN.TEST]. Joe wants to export the files in the subdirectory TEST to his UNIX machine, `ultra`.

3. Enter the following commands:

   ```
   $ TCPIP
   TCPIP> ADD PROXY BROWN /UID=27 /GID=58 /HOST=ultra
   TCPIP> MAP "/vmsdisk" DSA301:
   TCPIP> ADD EXPORT "/vmsdisk/brown/test" /HOST=ultra
   ```

   If you want to make the mapping permanent, enter a SET CONFIGURATION MAP command.

If users need to create files with case-sensitive names or names containing characters that do not conform to the OpenVMS syntax, you can enable name conversion, which gives users more file-naming flexibility without creating a container file system. Use the /OPTIONS=NAME_CONVERSION qualifier to the command ADD EXPORT to enable this option.

With the NAME_CONVERSION option set, users can create files and directories in an OpenVMS file system using names that do not conform to OpenVMS file-naming rules.

Note

If any client hosts had the file system mounted before the name conversion was enabled, they must dismount and remount for this feature to take effect.

For more information about file name conversion, see Appendix C.

20.9 Setting Up and Exporting a Container File System

A container file system is similar to a UNIX file system. When you create a container file system, you must specify an owner, using the /USER_NAME qualifier to the CREATE CONTAINER command.

When a container file system is created, a **container directory** is created, along with a **container file** in it. This container file provides compatibility with UNIX file storage attributes, such as file names, date and time stamps, UNIX protection masks, and UID ownership. If a container file system called NFS is created, it may look like the following example:

```
$ DIR DKA0:[NFS]

Directory DKA0:[NFS]

00012201$BFS.DIR;1   NFS.CONTAINER;1

Total of 2 files.
```

The files contained within the directory should not be manipulated directly within OpenVMS except in the case of incremental backups, which require a separate backup of the container file.

If the container file system is for the use of just one remote user, that user can be the owner. If it is for the use of several users, the owner should be a user whose UIC is mapped to UID=0/GID=1 (UNIX user root). In either case, the name set with this qualifier must already be registered in the proxy database. This user also becomes the owner of the internal root directory of the container.

To create a container file system on the NFS server, follow these steps:

1. Add a proxy entry for the owner of the container file system.

   ```
   TCPIP> ADD PROXY SYSTEM /UID=0 /GID=1 /HOST=*
   ```

2. Create an empty container file system on an OpenVMS volume, assign an owner, and set permissions.

   ```
   TCPIP> CREATE CONTAINER DSA101:[TEST] /USER_NAME=SYSTEM -
   _TCPIP> /ROOT_MODE=751 /HOST="june"
   ```

   The preceding example creates a container file system named TEST on device DSA101:. The user with a UID of 0 is assigned as owner. The permissions are assigned as follows:

   Owner: read, write, and execute (7)
   Group: read and execute (5)
   World: execute (1)

3. Map the OpenVMS volume on which the container file has been created.

   ```
   TCPIP> MAP "/test_dsk" DSA101:
   ```

   Note that it is important to map the underlying volume before mapping the container file system to make it available to the NFS server and the management control program. It is possible to use a volume both as an OpenVMS style file system and a container file system. If the disk was already in use as a OpenVMS style file system, it may already be mapped. In that case, you can skip this step.

4. Map the container file system to make it available to NFS client hosts. This mapping gives the file system its UNIX style name and UNIX style attributes. For example:

   ```
   TCPIP> MAP "/test" DSA101:[TEST]
   ```

   To make the mappings permanent, also use the SET CONFIGURATION MAP command.

5. If you do not already have proxies for the users, create them now. For example:

   ```
   TCPIP> ADD PROXY USER1 /UID=234 /GID=14 /HOST=*
   ```

6. In the root directory, create a top-level directory for each remote user. Be sure to specify directory ownership and set file permissions as needed for your environment. For example,

   ```
   TCPIP> CREATE DIRECTORY "/test/user1" /USER_NAME=USER1 /MODE=751 /HOST="june"
   ```

7. Export the root directory or the user top-level directories in the container file system. To export the root directory, enter:

   ```
   TCPIP> ADD EXPORT "/test" /HOST=*
   ```

   To export the user top-level directory, enter:

   ```
   TCPIP> ADD EXPORT "/test/user1" /HOST="june"
   ```

20.10 Maintaining a Container File System

This section reviews the commands you use to maintain and examine a container file system. Topics include:

- Displaying directory listings
- Copying files
- Removing links to a file or directory
- Deleting files
- Verifying the integrity of the file system
- Rebuilding the container file system

For complete command descriptions, see the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.

20.10.1 Displaying Directory Listings

Use the DIRECTORY command to display the contents of a directory. For example,

```
TCPIP> DIRECTORY "/path/name"
```

In this example, */path/name* is a valid UNIX directory specification that begins with a slash (/) and is enclosed in quotation marks.

The DIRECTORY command has the following qualifiers:

- /FULL specifies that a comprehensive list of information is displayed for each file displayed by the DIRECTORY command. The default provides a brief listing of the files in the directory.
- /VMS provides the corresponding OpenVMS file name for each file in the directory.

20.10.2 Copying Files into a Container File System

You cannot use the DCL command COPY to create files in a container file system, because the UNIX directory structure is fully contained in the corresponding container file. Instead, you must use the TCP/IP Services IMPORT command to copy a file from an OpenVMS directory into a container file system. Similarly, use the TCP/IP Services EXPORT command to copy a file from a container file system into an OpenVMS directory.

If the OpenVMS data file does not have the STREAM_LF record format, it will automatically be converted to STREAM_LF. Use the /NOCONVERT qualifier to prevent the conversion.

20.10.3 Removing Links to a File

A **link** is a directory entry referring to a file. A file can have several links to it. A link (hard link) to a file is indistinguishable from the original directory entry. Any changes to the file are independent of the link used to reference the file. A file cannot be deleted (removed) until the link count is zero.

Users can create multiple links to a file. A user sometimes creates a link to a file so that the file appears in more than one directory.

All links to a file are of equal value. If a file has two links and one link is removed, the file is still accessible through the remaining link. When the last existing link is removed (the link count is zero), the file is no longer accessible and is deleted.

Remove links to a file with the REMOVE FILE command. For example, to remove the link to a file named `letter` located at `/usr/smith`, enter the following command:

```
TCPIP> REMOVE FILE "/usr/smith/letter"
```

20.10.4 Removing Links to a Directory

Like UNIX files, UNIX directories have links to them. An empty directory is deleted when the last link to the directory is removed.

Remove links to a UNIX directory with the REMOVE DIRECTORY command. For example, to remove the directory `smith` at `/usr`, enter the following command:

```
TCPIP> REMOVE DIRECTORY "/usr/smith"
```

20.10.5 Deleting a Container File System

You can delete a container file system with all its directories and files by issuing the DELETE CONTAINER command. For example, to delete the UNIX container created on WORK1$:[GROUP_A], enter the following command:

```
TCPIP> DELETE CONTAINER WORK1$:[GROUP_A]
```

Use the UNMAP command to unmap the container file system before you delete it.

20.10.6 Verifying the Integrity of a Container File System

You may want to verify the integrity of your container file system under the following circumstances:

- If you are experiencing disk read or write errors or encountering problems backing up the container.
- If you are making copies or restoring files from a backup.

  The container file records the volume label and the Files–11 file identifiers of the actual files on the disk. If you copy the file system or change the volume label, you must run ANALYZE CONTAINER/REPAIR after you copy the files so that the file identifiers and volume label are corrected for the new location of the files.
- During system startup after a system failure.

You can use the ANALYZE CONTAINER command to check the integrity of your container file system. This command is similar in function to the DCL ANALYZE/DISK_STRUCTURE command.

Before analyzing the container file system, unmap it to prevent access to it during the analysis.

Note

The underlying OpenVMS file system must be mapped before you use the ANALYZE CONTAINER command.

For example, to verify the integrity of a container file system called /GroupA located in WORK1$:[GROUP_A], enter the following commands:

```
TCPIP> UNMAP "/GroupA"
TCPIP> MAP "/group_a" WORK1$:
TCPIP> ANALYZE CONTAINER WORK1$:[GROUP_A]
```

File system access to the container file is suspended while the container is being analyzed.

Table 20–2 lists the components of a container file system that are normally verified by the ANALYZE CONTAINER command.

Table 20–2 Container File System Components Analyzed

UNIX Item	OpenVMS Conceptual Equivalent	Description
Super block	Home block	Contains the basic information on the internal structuring of the container file.
Inode	File header	Each file or directory has an inode that contains information describing the file. The inode is a central definition of the file.
Directory	Directory	Contains the file names and directory hierarchy information. File name entries contain links to the inode information.
Bitmap	BITMAP.SYS	Contains the container file internal allocation information. Only one bitmap exists in the container file.

For a complete description of the ANALYZE CONTAINER command and its qualifiers, see the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.

20.10.7 Restoring a Container File System

For a typical image restore, follow normal OpenVMS procedures.

For a nonimage restore, an additional step is required after the restore. The Files–11 file identifiers are recorded in the container file. These must be updated by the TCP/IP management command ANALYZE CONTAINER /REPAIR.

This extra step is also required for an image restore if the save set is being restored with the /NOINITIALIZE qualifier to a volume with a different label or if it is being restored to a bound volume set that has a member that was added since the time of the image backup.

20.11 Setting Up NFS Security Controls

The NFS server and the OpenVMS operating system provide many levels of security controls you can use to protect your file systems. Section 20.1.3, Section 20.1.4, and Section 20.1.7 describe how the server uses the proxy and export databases to restrict client access, and how to use OpenVMS account privileges and file protections to control access to files and directories.

The NFS server provides additional security controls through the use of the `noproxy_enabled` attribute. You can set this attribute in the NFS server site-specific startup file SYS$STARTUP:TCPIP$NFS_SERVER_SYSTARTUP.COM.

The server uses this attribute while it is running. If the attribute is set, a proxy is not required for users attempting to access the NFS server. For more information about the NFS server attributes, see Table 20–3.

20.12 Modifying NFS Server Attributes

You can modify the way the NFS server works by specifying NFS server attributes using the SYSCONFIG command. The characteristics of the NFS server that you can modify include:

- Proxy security
- Default proxy UID
- Default proxy GID
- Maximum concurrent TCP threads
- Maximum concurrent UDP threads

To make permanent modifications:

1. If it does not already exist, create the file SYS$STARTUP:TCPIP$NFS_SERVER_SYSTARTUP.COM.
2. Include the SYSCONFIG command to set the variables. For example:

   ```
   $ SYSCONFIG -r nfs_server tcp_threads=20 udp_threads=40
   ```

3. Shut down and then restart the NFS server to make the changes take effect. For example:

   ```
   $ @SYS$STARTUP:TCPIP$NFS_SERVER_SHUTDOWN.COM
   $ @SYS$STARTUP:TCPIP$NFS_SERVER_STARTUP.COM
   ```

Future upgrades or installations will not overwrite the definitions in the TCPIP$NFS_SERVER_SYSTARTUP.COM file.

Modifying NFS server characteristics can affect NFS server performance. Be sure you understand the impact (review Section 20.15) before making any changes.

Table 20–3 describes the NFS server attributes.

Table 20–3 Modifying NFS Server Attributes

Attribute	Description
noproxy_enabled	Enables the use of the noproxy_uid and noproxy_gid attributes. If this attribute is not set to 1, proxies are required for server access. If the value is 0, files owned by a user that is not in the proxy database are assumed to be owned by UID=-2/GID=-2. If the value is 1, files owned by a user not in the proxy database are reported to be owned by the values of the noproxy_uid and noproxy_gid attributes.
noproxy_uid	Specifies the default UID when a user cannot be translated by the proxy.
noproxy_gid	Specifies the default GID when a user cannot be translated by the proxy.
tcp_threads	Specifies the number of concurrent TCP threads within the server. A value of zero will disable the TCP protocol.
udp_threads	Specifies the number of concurrent UDP threads within the server. This value must not be zero.

(continued on next page)

Table 20–3 (Cont.) Modifying NFS Server Attributes

Attribute	Description
vnode_age	Specifies the number of seconds in the time interval since the last file access request. The server keeps an activity timestamp for each opened file to help manage the open file cache. You can also modify this value with the /INACTIVITY qualifier to the SET NFS_SERVER command. The default setting for this variable is 120, or 2 minutes. Be careful not to set this value to a small interval; this might reduce performance.

20.13 Modifying File System Characteristics

The file SYS$STARTUP:TCPIP$NFS_SERVER_STARTUP.COM also defines a set of logical names that set the file system parameters. Table 20–4 describes these logical names.

Table 20–4 File System Logical Names

Logical Name	Description
TCPIP$CFS_CACHE_LOW_LIMIT	Defines the minimum size of the free buffer list. When the list is smaller than the value of this logical name, the file system starts to reclaim used buffers. The default is 4 buffers. The free buffer list needs at least 4 free buffers (not taken by cache). If the actual number of free buffers is less than TCPIP$CFS_CACHE_LOW_LIMIT, the used buffers are returned to the free list until the size of the free list reaches the value of TCPIP$CFS_CACHE_HIGH_LIMIT.
TCPIP$CFS_CACHE_HIGH_LIMIT	Defines the number of buffers the file system tries to keep in the free buffer list. The default is 8 buffers. See TCPIP$CFS_CACHE_LOW_LIMIT. In a busy server environment, setting this parameter higher is likely to improve performance.
TCPIP$CFS_CACHE_SIZE	Defines the maximum number of cache buffers to be allocated.
TCPIP$CFS_TRANSFERSIZE	Defines the optimum size (in bytes) of the data transferred between server and client on READ and WRITE operations. The default is 8K bytes (8192 bytes). This value is used in most NFS server implementations.
TCPIP$CFS_SHOW_VERSION	Sets the SHOW_VERSION logical name ON or OFF. If ON, the NFS server returns to the client file names with version numbers, even if there is only one version of the file. The default is OFF.
TCPIP$CFS_MODUS_OPERANDI	Defines various operating modes. Use only under the advice of your Compaq support representative.

(continued on next page)

Table 20–4 (Cont.) File System Logical Names

Logical Name	Description
TCPIP$CFS_FATAL_MESSAGES	Defines the terminal device to which the important error messages are directed, in addition to the normal error messages that are sent to the operator's console. The default is _OPA0:.

20.14 File Locking

TCP/IP Services supports a partial implementation of NFS network locking, which allows users to lock files. The software coordinates locks among remote users and between remote and local users. The file locking features is applicable regardless of whether the OpenVMS Record Management Services (RMS) is used. However, NFS does not coordinate network locking and RMS record locks.

Note

This version of NFS does not support byte-range locking. If a byte-range lock request is received, it is handled as a file lock request.

File locking is implemented using the Network Lock Manager (NLM) (also known remote procedure call, or RPC, `lockd`) and the Network Status Monitor (NSM) (also known as RPC `statd`). The NLM coordinates locks made by clients. The NSM recovers lock information in case the server or client fails. The NSM uses the NLM to keep the host list when the client or the server fails and reboots, as follows:

- If the client fails and reboots, it notifies the NSMs on its host list. In turn, the NSMs tell their local NLMs to free any locks held for that client.
- If the server fails, when it reboots it notifies the NSMs on each client host in its host list. In turn, the client NSMs tell their local NLMs to request again all the locks that were granted on their behalf by the server before it failed.

The NSM and the NLM are enabled if you select LOCKD/STATD in the TCPIP$CONFIG.COM configuration procedure. As a result, two processes are started when you start TCP/IP Services: TCPIP$LOCKD and TCPIP$STATD. The NLM can be configured with the following optional parameters:

- TCPIP$LOCKD_TIMEOUT_PERIOD specifies the timeout period (in seconds). This value defines the amount of time for the client to wait before retransmitting a lock request to which the server has not responded. The default setting is 5 seconds.
- TCPIP$LOCKD_GRACE_PERIOD specifies the grace period (in seconds). This value defines the amount of time the NLM will deny new lock requests after a failure while the NSM is recovering the lock status. The default setting is 15 seconds.

To set these parameters, create or edit the following file:

```
SYS$STARTUP:TCPIP$LOCKD_SYSTARTUP.COM
```

20.14.1 File Locking Service Startup and Shutdown

The file locking services can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$LOCKD_STARTUP.COM allows you to start up the LOCKD component independently.
- SYS$STARTUP:TCPIP$STATD_STARTUP.COM allows you to start up the STATD component independently.
- SYS$STARTUP:TCPIP$LOCKD_SHUTDOWN.COM allows you to shut down the LOCKD component independently.
- SYS$STARTUP:TCPIP$STATD_SHUTDOWN.COM allows you to shut down the STATD component independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$LOCKD_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the LOCKD component is started.
- SYS$STARTUP:TCPIP$LOCKD_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the LOCKD component is shut down.

20.15 Improving NFS Server Performance

This section provides information to help you identify and resolve problems and tune system performance.

20.15.1 Displaying NFS Server Performance Information

The SHOW NFS_SERVER command displays information about the running NFS server. You can use the information to tune NFS server performance.

You can enter SHOW NFS_SERVER for a specific client or host if it is listed in the proxy database. The counter information can be especially useful in determining the load on your system.

For more information about the SHOW NFS_SERVER command, refer the *Compaq TCP/IP Services for OpenVMS Management Command Reference*.

20.15.2 Displaying File System Information

The SHOW CFS command is useful for monitoring the distribution of the file system services and the consumption of system time by the various system services. See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for a detailed description of the SHOW CFS command.

20.15.3 Increasing the Number of Active Threads

The NFS server is an asynchronous, multithreaded process. This means that multiple NFS requests can be processed concurrently. Each NFS request is referred to as a thread. With increased server activity, client users may experience timeout conditions. Assuming the server host has the available resources (CPU, memory, and disk speed), you can improve server response by increasing the number of active threads. You do this by changing the value for the appropriate NFS server attributes, as described in Section 20.12.

The NFS server supports both TCP and UDP connections. You can control the maximum number of concurrent threads for each type of connection.

- To set the maximum number of TCP threads, set the `tcp_threads` attribute.
- To set the maximum number of UPD threads, set the `udp_threads` attribute.

Do not set the UDP maximum threads to zero. If you set the variable to zero, the protocol will be disabled.

If you increase the number of active threads, you should also consider increasing the timeout period on UNIX clients. You do this with the /TIMEOUT option to the TCP/IP Services MOUNT command.

If your clients still experience timeout conditions after increasing the number of active threads and the timout period on the client, you may need to upgrade your hardware.

20.15.4 OpenVMS SYSGEN Parameters That Impact Performance

The following OpenVMS SYSGEN parameters impact NFS server performance:

- CHANNELCNT

  The CHANNELCNT parameter sets the maximum number of channels that a process can use. Ensure that CHANNELCNT is set large enough to handle the total number of files accessed by all clients.

  Note

  The NFS server process is also limited by the FILLM of the TCPIP$NFS account's SYSUAF record. The effective value is the lower of the FILLM and CHANNELCNT values.

- ACP parameters

  The NFS server issues a large number of ACP QIO calls through CFS. Altering certain ACP parameters could yield better performance. If you have reliable disks, you may want to set the parameter ACP_DATACHECK to zero to avoid extra disk I/Os. Directory searching and file attribute management constitutes a majority of the ACP operations. Therefore, Compaq recommends that you increase parameters such as ACP_HDRCACHE, ACP_MAPCACHE, ACP_DIRCACHE, ACP_FIDCACHE, and ACP_DATACACHE.

- LOCK parameters

  The various lock manager parameters may need some alteration because CFS uses the lock manager extensively. A lock is created for each file system, each referenced file, and each data buffer that is loaded into the CFS cache.

- VIRTUALPAGECNT

  Maximum virtual size of a process in pages. The NFS server requires larger-than-normal amounts of virtual address space to accommodate structures and buffer space.

- WSMAX

  Maximum physical size of a process in pages. The larger the working set, the more pages of virtual memory that can remain resident. Larger values reduce page faults and increase the server's performance.

21

NFS Client

The Network File System (NFS) client software enables client users to access file systems made available by an NFS server. These files and directories physically reside on the remote (server) host but appear to the client as if they were on the local system. For example, any files accessed by an OpenVMS client — even a UNIX file — appear to be OpenVMS files and have typical OpenVMS file names.

This chapter reviews key concepts and describes:

- How to start up and shut down the NFS client (Section 21.2)
- How to register users in the proxy database (Section 21.3)
- How to mount files and directories (Section 21.4)

For information about the NFS server, see Chapter 20.

21.1 Key Concepts

Because the NFS software was originally developed on and used for UNIX machines, NFS implementations use UNIX file system conventions and characteristics. This means that the rules and conventions that apply to UNIX file types, file names, file ownership, and user identification also apply to NFS.

Because the TCP/IP Services NFS client runs on OpenVMS, the client must accommodate the differences between the two file systems, for example, by converting file names and mapping file ownership information. You must understand these differences to configure NFS properly and to successfully mount file systems from an NFS server.

The following sections serve as a review only. If you are not familiar with these topics, see the *DIGITAL TCP/IP Services for OpenVMS Concepts and Planning* guide for a more detailed discussion of the NFS implementation available with the TCP/IP Services software.

21.1.1 NFS Clients and Servers

NFS is a client/server environment that allows computers to share disk space and users to work with their files from multiple computers without copying them to the local system. Computers that make files available to remote users are NFS servers. Computers with local users accessing and creating remote files are NFS clients. A computer can be an NFS server or an NFS client, or both a server and a client.

Attaching a remote directory to the local file system is called **mounting** a directory. A directory cannot be mounted unless it is first **exported** by an NFS server. The NFS client identifies each file system by the name of its mount point on the server. The mount point is the name of the device or directory at the top of the file system hierarchy. An NFS device is always named DNFS*n*.

All files below the mount point are available to client users as if they reside on the local system. The NFS client requests file operations by contacting a remote NFS server. The server then performs the requested operation. The NFS client automatically converts all mounted directories and file structures, contents, and names to the format required by OpenVMS. For example, a UNIX file named `/usr/webster/.login` would appear to an OpenVMS client as `DNFS1:[USR.WEBSTER].LOGIN;1`

For more information on how NFS converts file names, see Appendix C.

21.1.2 Storing File Attributes

The OpenVMS operating system supports multiple file types and record formats. In contrast, NFS and UNIX systems support only byte-stream files, seen to the OpenVMS client as sequential STREAM_LF files.

This means the client must use special record handling to store and access non-STREAM_LF files. The OpenVMS NFS client accomplishes this with attribute description files (ADFs). These are special companion files the client uses to hold the attribute information that would otherwise be lost in the translation to STREAM_LF format. For example, a SET FILE/NOBACKUP command causes the client to create an ADF, because NFS has no concept of this OpenVMS attribute.

21.1.2.1 Using Default ADFs

The client provides default ADFs for files with the following extensions: .EXE, .HLB, .MLB, .OBJ, .OLB, .STB, and .TLB. (The client does not provide ADFs for files with the .TXT and .C extensions, because these are STREAM_LF.) The client maintains these ADFs on the server.

For example, SYS$SYSTEM:TCPIP$EXE.ADF is the default ADF for all .EXE type files. When you create .EXE files (or if they exist on the server), they are defined with the record attributes from the single default ADF file. The client refers only to the record attributes and file characteristics fields in the default ADF.

21.1.2.2 How the Client Uses ADFs

By default, the client uses ADFs if they exist on the server. The client updates existing ADFs or creates them as needed for new files. If you create a non-STREAM_LF OpenVMS file or a file with access control lists (ACLs) associated with it on the NFS server, the NFS client checks to see whether a default ADF can be applied. If not, the client creates a companion ADF to hold the attributes.

The client hides these companion files from the user's view. If a user renames or deletes the orginal file, the client automatically renames or deletes the companion file. However, if a user renames or deletes a file on the server side, the user must also rename the companion file; otherwise, file attributes are lost.

You can modify this behavior with the /NOADF qualifier to the MOUNT command. The /NOADF qualifier tells the client to handle all files as STREAM_LF unless a default ADF matches. This mode is only appropriate for read-only file systems because the client cannot adequately handle application-created files when /NOADF is operational.

21.1.2.3 Creating Customized Default ADFs

You can create customized default ADFs for special applications. To do so:

1. On the client, create a special application file that results in creating an ADF on the server. Suppose that application file is called TEST.GAF.
2. On the server, check the listing for the newly created file. For example:

```
> ls -a
.
..
.$ADF$test.gaf;1
test.gaf
```

   Note that the ADF (.ADFtest.gaf;1) was created with the data file (TEST.GAF).

3. On the server, copy the ADF file to a newly created default ADF file on the client. For example:

```
> cp .\$ADF\$test.gaf\;1 gaf.adf
```

   Note that the backslashes (\) are required for entering the UNIX system nonstandard dollar sign ($) and semicolon (;) symbols.

4. On the client, copy the new default ADF file to the SYS$SYSTEM directory. For example:

```
$ COPY GAF.ADF SYS$COMMON:[SYSEXE]TCPIP$GAF.ADF
```

5. Dismount all the NFS volumes and mount them again. This starts another NFS ancillary control process (ACP) so that the newly copied default ADF file can take effect.

21.1.3 How the NFS Client Authenticates Users

Both the NFS server and NFS client use the proxy database to authenticate users. The proxy database is a collection of entries used to register user identities. To access file systems on the remote server, local users must have valid accounts on the remote server system.

The proxy entries map each user's OpenVMS identity to a corresponding NFS identity on the server host. When a user initiates a file access request, NFS checks the proxy database before granting or denying access to the file.

The proxy database is an index file called TCPIP$PROXY.DAT. If you use the configuration procedure to configure NFS, this empty file is created for you. You populate this file by adding entries for each NFS user. See Section 21.3 for instructions on how to add entries to the proxy database.

Note

The configuration procedure for the NFS server creates a nonprivileged account with the user name TCPIP$NOBODY. You may want to add a proxy record for the default user (-2/-2) that maps to the TCPIP$NOBODY account.

21.1.4 How the Client Maps User Identities

Both OpenVMS and UNIX based systems use identification codes as a general method of resource protection and access control. Just as OpenVMS employs user names and UICs for identification, UNIX identifies users with a user name and a user identifier (UID) and group identifier (GID) pair. Both UIDs and GIDs are used to identify a user on a system.

The proxy database contains entries for each user wanting to access files on a server host. Each entry contains the user's local OpenVMS account name, the UID/GID pair that identifies the user's account on the server system, and the name of the server host. This file is loaded into dynamic memory when the NFS client starts. Whenever you modify the UID/GID to UIC mapping, you must restart the NFS client software by dismounting and remounting all the client devices. (Proxy mapping always occurs even when operating in OpenVMS to OpenVMS mode.)

The only permission required by the UNIX file system for deleting a file is write access to the last directory in the path specification.

You can print a file that is located on a DNFS*n*: device. However, the print symbiont, which runs as user SYSTEM, opens the file only if it is world readable or if there is an entry in the proxy database that allows read access to user SYSTEM.

21.1.4.1 Default User

You can associate a client device with a default user by designating the user with the /UID and /GID qualifiers to the MOUNT command. If you do not specify a user with the /UID and /GID qualifiers, NFS uses the default user –2/–2. If the local user or the NFS client has no proxy for the host serving a DNFS device, all operations performed by that user on that device are seen as coming from the default user (–2/–2).

To provide universal access to world-readable files, you can use the default UID instead of creating a proxy entry for every NFS client user.

Compaq strongly recommends that, for any other purposes, you provide a proxy with a unique UID for every client user. Otherwise, client users may see unpredictable and confusing results when they try to create files.

21.1.5 How the Client Maps UNIX Permissions to OpenVMS Protections

Both OpenVMS and UNIX based systems use a protection mask that defines categories assigned to a file and the type of access granted to each category. The NFS server file protection categories, like those on UNIX systems, include: `user`, `group` and `other`, each having read (`r`), write (`w`), or execute (`x`) access. The OpenVMS categories are SYSTEM, OWNER, GROUP, and WORLD. Each category can have up to four types of access: read (R), write, (W), execute (E), and delete (D). The NFS client handles file protection mapping from server to client.

OpenVMS delete access does not directly translate to a UNIX protection category. A UNIX user can delete a file as long as he or she has write access to the parent directory. The user can see whether or not he or she has permissions to delete a file by looking at the protections on the parent directory. This design corresponds to OpenVMS where the absence of write access to the parent directory prevents users from deleting files, even when protections on the file itself appear to allow delete access. For this reason, the NFS client always displays the protection mask of remote UNIX files as permitting delete access for all categories of users.

Since a UNIX file system does not have a SYSTEM protection mask (the superuser has all permissions for all files) the NFS client displays the SYSTEM as identical to the OWNER mask.

21.1.6 Guidelines for Working with DNFS Devices

The following list summarizes the guidelines and restrictions associated with DNFS devices:

- BACKUP and RESTORE operations

  The OpenVMS NFS client does not emulate the on-disk structure of actual OpenVMS disks. Therefore, applications that need direct knowledge of the OpenVMS on-disk structure, such as image backup and restore, work differently with DNFS*n*: volumes than with other volumes.

- File identification

  The NFS client constructs OpenVMS file identifiers (FIDs) dynamically. The remote NFS server does not store them. Each NFS client constructs its own FIDs, possibly leading to different FIDs of the same file for different NFS clients.

- Disk quotas

  Disk quotas for OpenVMS disks are not applicable to DNFS*n*: disks.

21.1.7 How NFS Converts File Names

Because NFS uses UNIX style syntax for file names, valid OpenVMS file names may be invalid on the NFS server and vice versa. The NFS software automatically converts file names to the format required by either the client or the server. (NFS always converts file names even when both the NFS client and the NFS server are OpenVMS hosts.)

All name-mapping sequences on the OpenVMS client begin with the dollar sign ($) escape character. Appendix C lists the rules that govern these conversions and provides a list of character sequences, server characters, and octal values used for NFS name conversion.

21.2 NFS Client Startup and Shutdown

The NFS client can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$NFS_CLIENT_STARTUP.COM allows you to start up the NFS client independently.
- SYS$STARTUP:TCPIP$NFS_CLIENT_SHUTDOWN.COM allows you to shut down the NFS client independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$NFS_CLIENT_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NFS client is started.

  For example, use this file to store systemwide MOUNT commands.

- SYS$STARTUP:TCPIP$NFS_CLIENT_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked immediately before the NFS client is shut down.

21.3 Registering Users in the Proxy Database

Users on your client host must have corresponding accounts on the NFS server host. After making sure client users have appropriate accounts, you must register them with the proxy database. The NFS client, the NFS server, and the PC-NFS daemon all use the proxy database.

If you use TCPIP$CONFIG to configure NFS, the index file TCPIP$PROXY.DAT is created for you. This file is empty until you populate it with proxy entries. If you do not use the configuration procedure, use the CREATE PROXY command to create the empty database file. The file TCPIP$PROXY.DAT resides in the SYS$COMMON:[SYSEXE] directory by default. You can change the location of the proxy database by redefining the logical name TCPIP$PROXY. (You can also create a proxy database file from a UNIX formatted `/etc/password` file by using the CONVERT/VMS PROXY command.)

Use the following TCP/IP management commands to manage the proxy database:

- ADD PROXY
- REMOVE PROXY
- SHOW PROXY

For example:

```
TCPIP> ADD PROXY username /NFS=type /UID=n /GID=n /HOST=host_name
```

Changes in the proxy database take effect only after you dismount all DNFS*n*: devices and remount them. An exception is DNFS0:, which is present if the NFS client driver is loaded and cannot be mounted or dismounted.

Each entry in the proxy database has the fields that are listed in Table 21–1.

Table 21–1 Required Fields for NFS Proxy Entries

Field	Meaning
OpenVMS user name	Name of the NFS user's OpenVMS account
Type	Direction of NFS communication allowable to the user. Specify one of the following: • O (outgoing). Used by the NFS client. • N (incoming). Used by the NFS server. • ON (outgoing and incoming). Used by both client and server. • D (dynamic). Entry is loaded in the server's dynamic memory. When the NFS server starts, it creates a copy of the proxy database in dynamic memory. (If the account does not exist or the account is disabled, the entry for the account will be missing from dynamic memory.)

(continued on next page)

Table 21–1 (Cont.) Required Fields for NFS Proxy Entries

Field	Meaning
UID/GID pair	Remote identity of the user. Required even if both client and server are OpenVMS hosts.
Remote host name	Name of the remote host, which is one of the following: • Remote client of the local NFS server • Remote server for the local NFS client • Both • Wildcard (*) for all hosts

To add a user name to the proxy database, take the following steps:

1. For each NFS user, obtain the OpenVMS user name from the OpenVMS user authorization file (UAF). If a user name is not in the UAF, use the OpenVMS Authorize utility to add it.
2. Obtain the UID/GID pair for each NFS user from the `/etc/password` file on the NFS server.
3. Enter SHOW PROXY.
4. Enter ADD PROXY for each NFS user you want to add to the proxy database. For example:

   ```
   TCPIP> ADD PROXY GANNET /NFS=(OUTGOING,INCOMING) /UID=1111 /GID=22 /HOST=CLIENT1
   ```

5. Reenter SHOW PROXY to confirm the new information.

The following illustrates a portion of a proxy database file:

```
VMS User_name     Type      User_ID    Group_ID   Host_name
GANNET            OND          1111          22   CLIENT1, client1
GEESE             OND          1112          22   *
GREBE             OND          1113          22   client1, client2
GROUSE            OD           1114          23   client3
GUILLEMOT         OD           1115          23   client3
GULL              OD           1116          23   client4
```

21.4 Mounting Files and Directories

Attaching remote files and directories exported by an NFS server is called mounting. The NFS client identifies each file system by the name of its mount point on the server. The client provides the following TCP/IP management commands:

- MOUNT
- SHOW MOUNT
- DISMOUNT

For example:

```
TCPIP> MOUNT mount_point /HOST="host" /PATH="/path/name"
```

Note

By default, a mount is considered a system mount and privileges are required unless the /SHARE qualifier is used. See Section 21.4.1 for information on user-level mounting.

When you issue a MOUNT command, the NFS client creates a new DNFS device and mounts the remote file system onto it. For example, the following command mounts, onto local device DNFS2:, the remote directory /usr/users/curlew, which physically resides on NFS server loon.

```
TCPIP> MOUNT DNFS2: /HOST="loon" /PATH="/usr/users/curlew"
```

After entering the command, a confirmation message such as the following is displayed:

```
%DNFS-S-MOUNTED, /users/curlew mounted on DNFS2:[000000]
```

If you specify DNFS0 in a mount command, the client selects the next available unit number for you, for example:

```
MOUNT DNFS0:/HOST="loon" /PATH="/usr/curlew"
%DNFS-S-MOUNTED, /usr/curlew mounted on DNFS3:[000000]
```

Qualifiers to the MOUNT command let you modify the way a traditional mount occurs. For example, you may specify background mounting, modify existing mounts, or hide subdirectories from view. See the following sections for more information:

- User-level mounting (Section 21.4.1)
- Automounting (Section 21.4.2)
- Background mounting (Section 21.4.3)
- Overmounting (Section 21.4.4)
- Occluded mounting (Section 21.4.5)

See the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual for a complete list of MOUNT options and command qualifiers.

21.4.1 User-Level Mounting

The NFS client supports shared mounting by using the /SHARE qualifier with the MOUNT command. Any user can mount a file system using the /SHARE qualifier—SYSNAM or GRPNAM privileges are not required. The /SHARE qualifier places the logical name in the job logical name table and increments the volume mount count, regardless of the number of job mounts. When the job logs out, all job mounts are dismounted, which causes the volume mount count to be decremented.

The following example illustrates how to specify a shared mount:

```
TCPIP> MOUNT DNFS1: /HOST=BART /PATH="/DKA100/ENG"
TCPIP> MOUNT DNFS1: /HOST=BART /PATH="/DKA100/ENG" /SHARE
```

This mount request increments the mount count by 1. You must specify the /SHARE qualifier with the same host name and path as used in the initial mount to ensure that the mount is seen as a shared mount instead of as a new mount request.

With a shared mount, the mount requests increment the mount count by 1 under the following circumstances:

- With an initial /SYSTEM or /GROUP mount.
- With a DCL command MOUNT/SHARE or a TCP/IP management command MOUNT/SHARE that completes without an error. (This contrasts with overmount, where the previous mounting point is dismounted. This condition can increment or decrement the mount count or leave it unchanged.)

In this way, if the main process of the job logs out, the job mount is deallocated, and the volume mount count decrements by 1 (if zero, the device is dismounted). OpenVMS handles dismounting differently based on whether you use the TCP/IP management command DISMOUNT or the DCL command DISMOUNT. These differences are as follows:

- With the TCP/IP command DISMOUNT, the NFS ACP dismounts one or (in the case of /ALL) more mount points. If the mount point being dismounted is the only or last one for the device, the device is dismounted for all users who mounted it, and the mount count is decremented to zero. If more that one mount point exists, the mount point is dismounted along with a specifically shared mount.
- With the DCL command DISMOUNT, the OpenVMS operating system checkes for job mounts first. If a job mount for the specified device exists, the /JOB mount is dismounted, any logical name associated with the /JOB mount is deallocated, and the mount count is decremented by one. If no JOB mount exists, OpenVMS checks for /SYSTEM and /GROUP mounts. If one exists, and the user has the required privilege, the /SYSTEM or /GROUP mount is dismounted, any associated logical name is deallocated, and the mount count is decremented by 1. No mount points are dismounted until the mount count reaches zero.

  If the user does not have the required SYSNAM or GRPNAM privilege, the following error message is returned:

  ```
  SYSTEM-F-NO-PRIVILEGE, operation requires privilege
  ```

  If no /SYSTEM or /GROUP mount exists, the following error message is returned:

  ```
  %DISM-W-CANNOTDMT, NFSn: cannot be dismounted
  %SYSTEM -F -DEVNOTMOUNT, device is not mounted
  ```

Consider the mount counts in the following sample MOUNT/DISMOUNT sequence:

1. `TCPIP> MOUNT DNFS1:/HOST=BART /PATH="/DKA0/ENG"/`

   Mount count: 1 system mount, not incremented

2. `TCPIP> MOUNT DNFS1:[A] /HOST=BART /PATH="/DKA0/ENG" /SHARE`

   Mount count: 2 (incremented)

3. `$ MOUNT/SHARE DNFS1:`

   Mount count: 3 (incremented)

4. `TCPIP> MOUNT DNFS1:[B] /HOST=MARGE /PATH="DKA0/TEST"`

   Mount count: 3 (system mount, not incremented)

5. `TCPIP> DISMOUNT DNFS1:[A]`

   Mount count: 2

6. `$ DISMOUNT DNFS1:`

   Mount count: 1 (removed mount in example 3, decremented)

7. `$ DISMOUNT DNFS1:`

   Mount count: 0 (removed mount in example 4, decremented)

The original mount for BART "/ENG" on DNFS1:[A], along with its shared mount, is dismounted. The subsequent DISMOUNT commands dismount examples 3 and 4, leaving nothing mounted.

21.4.2 Automounting

Automounting allows you to mount a remote file system on an as-needed basis. This means that the client automatically and transparently mounts a remote server path as soon as the user accesses the path name.

Automounting is convenient for file systems that are inactive for large periods of time. When a user on a system invokes a command to access a remote file or directory, the automount daemon mounts the file and keeps it mounted as long as the user needs it. When a specified amount of time elapses without the file being accessed, it is dismounted. You can specify an inactivity period (5 minutes is the default), after which the software automatically dismounts the path.

You specify automounting and an inactivity interval with the qualifier /AUTOMOUNT=INACTIVITY:*OpenVMS_delta_time*.

The inactivity interval is the maximum inactive period for the mount attempt. When this period expires, the NFS client dismounts the path name as described below.

In this example, the client automounts directory `/usr/webster` residing on host `robin` onto the OpenVMS mount point DNFS67:. When it references the path name, the client keeps the path mounted unless it reaches an inactive period of 10 minutes, after which it dismounts the file system. With subsequent references, the client remounts the file system. For example:

```
TCPIP> MOUNT DNFS67: /HOST="robin" -
_TCPIP> /PATH="/usr/webster" /AUTOMOUNT=INACTIVITY=00:10:00
```

21.4.3 Background Mounting

Background mounting allows you to retry a file system mount that initially failed. For example, you may have set mount points in your system startup command file so they are automatically mounted every time your system reboots. In this scenario, if the server is unavailable (because, for example, the server is also rebooting), the mount requests fail. With background option set, the client continues to try the mount after the initial failure. The client continues trying up to 10 times at 30-second intervals (default) or for the number of retries and interval you specify.

If you specify background mounting, you should also use the /RETRIES qualifier with a small nonzero number. This qualifier sets the number of times the transaction itself should be retried. Specify background mounting, along with the desired delay time and retry count parameters, with the qualifier /BACKGROUND=[DELAY:*OpenVMS_delta_time*,RETRY:*n*].

For example, the following command attempts to mount in background mode, on local device DNFS4:, the file system `/flyer`, which physically resides on host `migration`. If the mount fails, the NFS client waits 1 minute and then retries the connection up to 20 times. For example:

```
TCPIP> MOUNT DNFS4: /HOST="migration" /PATH="/flyer" -
_TCPIP> /BACKGROUND=(DELAY:00:01:00, RETRY:20) /RETRIES=4
```

If you use the /BACKGROUND qualifier, Compaq strongly recommends that you also use the /RETRIES qualifier specifying a nonzero value. If you use the default value for /RETRIES (zero), the first mount attempt can never complete except by succeeding, and the process doing the mount will hang until the server becomes available.

21.4.4 Overmounting

Overmounting allows you to mount another path onto an existing mount point. Specify overmounting with the /FORCE qualifier. The client dismounts the original mount point and replaces it with a new one.

Mounting a higher or lower directory level in a previously used path is also an overmount. For example, an overmount occurs when you execute two MOUNT commands in the following order:

```
TCPIP> MOUNT DNFS123:[USERS.MNT] /HOST="robin" /PATH="/usr"

%DNFS-S-MOUNTED, /usr mounted on _DNFS123:[USERS.MNT]

TCPIP> MOUNT DNFS123:[USERS.MNT] /HOST="robin" /PATH="/usr/tern" /FORCE

%DNFS-S-REMOUNTED, _DNFS123:[USERS.MNT] remounted as /usr/tern on ROBIN
```

The second MOUNT command specifies a lower level in the server path. This constitutes another path name and qualifies for an overmount.

21.4.5 Occluded Mounting

Occluded mounting allows you to mount a file system onto a client mount point that is higher or lower in the directory structure than an existing, active mount. This is different from overmounting because dismounting does not occur. Instead, the client occludes (hides from view) the subdirectories that are added to or dropped from the original mount specification when you perform a directory listing.

Specify the /FORCE qualifier with an occluded mount.

In the following example, the mount point specification was backed up one subdirectory from the previous one. If you enter the SHOW MOUNT command, both mounts are visible. However, if you enter DIRECTORY for DNFS2:[USERS.SPARROW], [.MNT] is no longer visible. To make this subdirectory visible again, issue the DISMOUNT command to dismount DNFS2:[USERS.SPARROW].

```
TCPIP> MOUNT DNFS2:[USERS.SPARROW.MNT] /HOST="birdy" /PATH="/usr"

%DNFS-S-MOUNTED, /usr mounted on _DNFS2:[USERS.SPARROW.MNT]

TCPIP> MOUNT DNFS2:[USERS.SPARROW] /HOST="birdy" /PATH="/usr" /FORCE

%DNFS-S-MOUNTED, /usr mounted on _DNFS2:[USERS.SPARROW]
-TCPIP-I-OCCLUDED, previous contents of _DNFS2:[USERS.SPARROW] occluded
```

The following example shows a mount of UNIX directory /usr to the OpenVMS device and directory DNFS3:[0,0].

On the UNIX host, the directory listing looks like this:

```
unix% ls
grebe wings pratincole
```

To do the mount, enter:

```
$ TCPIP MOUNT DNFS3: /HOST="unix" /PATH="/usr"
```

To check that the mount succeeded, enter:

```
$ TCPIP SHOW MOUNT DNFS3: /FULL
.
.
.
```

On the OpenVMS host, the directory listing looks like this:

```
$ DIRECTORY [0,0]

Directory DNFS3:[000,000]

GREBE.DIR;1  WINGS.DIR;1   PRATINCOLE.DIR;1

Total of 3 files.
```

Part 6

Configuring Printing Services

Part 6 describes how to set up and manage the printing services available with TCP/IP Services, and includes the following chapters:

- Chapter 22, Setting Up and Managing the LPR/LPD Print Service, describes how to set up LPR/LPD, providing access to local and remote print queues.
- Chapter 23, Setting Up and Managing TELNETSYM, describes how to set up and manage the TELNET print symbiont (TELNETSYM). TELNETSYM provides remote print services that enable the use of standard OpenVMS printing features not available with LPR/LPD.
- Chapter 24, Setting Up PC-NFS, describes how to set up and manage printing for PC-NFS client users.

22

Setting Up and Managing the LPR/LPD Print Service

The LPR/LPD service allows other network hosts to access printers on the server system and provides local access to printers on remote hosts. Remote print server and the client hosts must run Version 4.2 or later of the Berkeley Software Distribution line printer spooler software (lpd) to interoperate with TCP/IP Services LPR/LPD.

This chapter reviews key concepts and describes:

- How to configure the LPR/LPD print service (Section 22.2)
- How to configure printers (Section 22.3)
- How to manage LPD server queues (Section 22.4)
- How to control access to local LPD server queues (Section 22.5)
- How to enable LPR/LPD OPCOM messages (Section 22.6)
- How to use OpenVMS flag page options with LPR/LPD (Section 22.7)
- How to solve LPR/LPD problems (Section 22.8)

22.1 Key Concepts

The LPR/LPD service has both a client component (LPR) and a server component (LPD), both of which are partially included in an OpenVMS queue symbiont. The client is activated when you use one of the following commands:

- PRINT—to submit a print job to a remote printer whose queue is managed by the LPD symbiont.
- LPRM—to remove (cancel) a pending print job previously spooled.
- LPQ—to view the queue of pending jobs for a remote printer.

See the *DIGITAL TCP/IP Services for OpenVMS User's Guide* for general information about using these commands.

The server is activated when a remote user submits a print job to a printer configured on the OpenVMS server. The LPD server consists of two components:

- LPD receiver—a process that handles the incoming request from the remote system over the network. It copies the control file (CF) and data file (DF) representing the print job to the requested printer's LPD spool directory, and places the control file on the print queue for further processing. The receiver also handles LPQ and LPRM functions from remote clients.
- LPD symbiont—which parses the print job's control file and submits the data files to the designated local printer's print queue.

The same LPD symbiont image is used for both client and server. It acts as the client on queues set up for remote printers, and it acts as the server on the local LPD queue.

The LPD uses the printcap database to process print requests. The printcap database, located in SYS$SPECIFIC:[TCPIP$LPD]:TCPIP$PRINTCAP.DAT, is an ASCII file that defines the print queues. The printcap entries are similar in syntax to the entries in a UNIX `/etc/printcap` file.

Use the printer setup program LPRSETUP to configure or modify printers. The setup program creates spool directories and log files based on the information you supply. Section 22.3 describes how to use the printer setup program to configure printers.

22.2 Configuring LPR/LPD

If you use the configuration procedure to configure LPR/LPD, the procedure completes the following tasks:

- Adds the LPD service to the services database.
- Adds the TCPIP$LPD account to the SYSUAF.DAT database.
- Creates the directory SYS$SPECIFIC:[TCPIP$LPD] for the TCPIP$LPD account.
- Enables both client and server components.

After you start the service, the logical names listed in Table 22–1 are defined.

Table 22–1 LPD Logical Names

Logical Name	Description
TCPIP$LPD_CLIENT_ENABLE	Enables the LPD client.
TCPIP$LPD_PRINTCAP	Specifies the database that maps queues and makes queues available.
TCPIP$LPD_SPOOL	Specifies the main spooling directory.
TCPIP$LPD_LOGFILE	Specifies the name of the log file for the TCPIP$LPD_QUEUE, which precedes all inbound jobs.
TCPIP$LPD_STREAM_PASSALL	This version of TCP/IP Services adds extra line feed characters to files with embedded carriage control. This logical preserves the behavior of previous versions of TCP/IP Services and is useful when your users print from Compaq *PATHWORKS* Client software.

(continued on next page)

Table 22–1 (Cont.) LPD Logical Names

Logical Name	Description
TCPIP$LPD_KEEPALIVE	The KEEPALIVE timer is used to periodically check the other end of a link that appears to be idle. The purpose of the time is to detect when a remote host has failed or has been brought down, or when the logical connection has been broken. Set this logical using the following command: `$ DEFINE/SYSTEM TCPIP$LPD_KEEPALIVE` This logical name is not used by the server; it is used by LPD. If you are changing this logical name, there is no need to restart TCP/IP Services. If this logical is defined, the KEEPALIVE function is enabled. By default, the KEEPALIVE timer is disabled. Broken connections will be detected only if the relevant application sends data.
TCPIP$LPD_PROBETIME	The PROBE timer specifies: 1. When establishing an initial connection, the number of seconds TCP/IP Services will wait for a response before a timeout occurs. The time is active regardless of whether the TCPIP$LPD_KEEPALIVE logical is set. 2. The length of time (in seconds) allowed to pass before TCP/IP Services checks an idle connection. This requires that the TCPIP$LPD_KEEPALIVE logical be set. You control the PROBE timer using the following command: `$ DEFINE/SYS TCPIP$LPD_PROBE` *x* In this command, *x* specifies the number of seconds to wait before timing out the connection. The value of the PROBE timer must always be less than or equal to the value of the DROP timer. The default value for the PROBE timer is 75 seconds.

(continued on next page)

Table 22–1 (Cont.) LPD Logical Names

Logical Name	Description
TCPIP$LPD_DROPTIME	The DROP timer indicates how long (in seconds) that a connection should be maintained (after repeated timeouts) before closing the connection. The DROP timer is in effect only after the link has been established, and it takes effect only if the TCPIP$KEEPALIVE logical is set. You control the DROP timer using the following command: `$ DEFINE/SYS TCPIP$LPD_DROP x` In this command, *x* specifies the number of seconds to maintain the connection before closing it. The default value for the DROP timer is 300 seconds.
TCPIP$LPD_1ST_VFC_PREFIX_SPECIAL	LPD can insert an extra line feed character at the beginning of print files. Set this logical to suppress this action.
TCPIP$LPD_VMS_FLAGPAGES	Enables the OpenVMS flag page print options, as described in Section 22.7.
TCPIP$LPD_PS_EXT	Control Compaq PrintServer extension support. By default, PrintServer extensions are supported by LPD. To disable PrintServer extension support on the system, enter the following command: `$ DEFINE/SYSTEM TCPIP$LPD_PS_EXT NON_PS` To enable PostScript extensions, specify LPS as the value of this logical name. For more information about configuring PrintServer extension support for a printer, see Section 22.3.
TCPIP$LPD_DEBUG	Writes diagnostics to the LPD queue log file. Applies to outbound jobs (LPD client) and to inbound jobs (LPD server) that are processed by the LPD symbiont controlling the local print queue. See the description of the TCPIP$LPD_RCV logical for more information.

(continued on next page)

Table 22–1 (Cont.) LPD Logical Names

Logical Name	Description

TCPIP$LPD_RCV	Writes diagnostics to the receiver log file TCPIP$LPD_RCV_LOGFILE.LOG. Applies to inbound jobs (LPD server) from the time they are received from the remote host over the network to the time they are queued to the local print queue for processing by the LPD print symbiont.
	TCPIP$LPD_DEBUG and TCPIP$LPD_RCV are bit-mapped values. The low-order three bits turn on all diagnostics generated by either the sender or the receiver.
	To define these logical names, set the following bits in the value:
	• Bit 0 indicates minimal debugging information.
	• Bit 1 indicates an intermediate amount of debugging information.
	• Bit 2 indicates the full amount of debuggging information available.
	• Bit 3 logs the actual data sent and received over the network.
	If you set the fourth bit, the LPD symbiont logs each buffer that it sends over the TCP/IP link, and the LPD receiver logs each buffer that it receives from the TCP/IP link. The log files let you see exactly what the LPD is sending (for outbound jobs) and receiving (for inbound jobs).
	To set the fourth bit, enter:
	`$ DEFINE /SYSTEM LPD_RCV 8`
	`$ DEFINE /SYSTEM LPD_DEBUG 8`
	For more information about using these logical to solve printing problems, see Section 22.8.

22.2.1 LPD Server Startup and Shutdown

The LPD server can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$LPD_STARTUP.COM allows you to start up the LPD server independently.
- SYS$STARTUP:TCPIP$LPD_SHUTDOWN.COM allows you to shut down LPD server independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$LPD_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the LPD server is started.
- SYS$STARTUP:TCPIP$LPD_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the LPD server is shut down.

22.3 Configuring Printers

This section describes how use the printer setup program, SYS$SYSTEM:TCPIP$LPRSETUP.EXE, to configure a printer directly connected to your computer. Similar to the UNIX `/usr/sbin/lprsetup` utility, you can also use this program to modify a printer's configuration or to remove a printer.

Before running the printer setup program, you need the following information for each printer you want to configure:

- Printer name, including all synonyms (aliases)
- Printer type (local or remote)
- Host and printer name for the remote printer
- Spool directory
- Error log directory

The printer setup program performs the following:

- Creates or edits the existing printcap database.
- Creates a spooling directory.
- Creates an error log file.
- Prompts you to modify previously selected symbols.

Table 22–2 describes the LPRSETUP commands.

Table 22–2 LPRSETUP Commands

Command	Description
add	Adds a printer name. The printer name is the name of a LPD client print queue that users can specify in the /QUEUE qualifier to the PRINT command.
delete	Removes an existing printer from your configuration.
view	Displays the contents of the current printcap database.
help	Displays online help about the LPRSETUP program.
exit	Exits from the LPRSETUP program.

You can abbreviate any command option with its initial letter. Enter information at each prompt, or press Return (or Enter) to accept the default. Enter a question mark (?) to obtain a description of the information requested at each prompt.

The following example shows how to use the printer setup program to configure a printer named LOCAL1:

```
$ RUN SYS$SYSTEM:TCPIP$LPRSETUP

TCPIP Printer Setup Program

Command  < add delete view help exit >: add
Adding printer entry, type '?' for help.

Enter printer name to add : LOCAL1
Enter the FULL name of one of the following printer types:
remote local : local
Enter printer synonym:

Enter full file specification for spool directory
SPOOLER DIRECTORY 'sd' :  [SYS$SPECIFIC:[TCPIP$LPD.LOCAL1]] ?
Enter full file specification for printer log file.
printer error log file 'lf' [SYS$SPECIFIC:[TCPIP$LPD]LOCAL1.LOG] ?
Enter the name of the printcap symbol you want to modify.  Other
valid entry is :
        'q'     to quit (no more changes)

The names of the printcap symbols are:

 sd for the printer spool directory
 lf for the printer error log file
 lp for the name of the local printer
 ps for the LPD PrintServer extensions flag
 rm for the name of the remote host
 rp for the name of the remote printer
 fm for the printer form field
 pa for the /PASSALL flag
 nd for the /NODELETE flag
 cr for the cr flag
 p1-p8 for the /PARAMETER=(p1,...,p8) field

Enter symbol name: q

                   Symbol  type  value
                   ------  ----  -----
Error log file    :  lf     STR   /SYS$SPECIFIC/TCPIP$LPD/LOCAL1.LOG
Printer Queue     :  lp     STR   LOCAL1
Spool Directory   :  sd     STR   /SYS$SPECIFIC/TCPIP$LPD/LOCAL1

Are these the final values for printer LOCAL1 ? [y]

Adding comments to printcap file for new printer, type '?' for help.
Do you want to add comments to the printcap file [n] ? :

        ********************************************
        * TCPIP$LPD_SYSTARTUP.COM   TCPIP$LPD_PRINTCAP*
        *    and TCPIP$LPD_SYSHUTDOWN.COM             *

        *  have been updated for this printer         *
        *                                             *
        * Set up activity is complete for this printer*
        ********************************************


Command  < add delete view help exit >: exit
```

The following example shows how to use the printer setup program to remove a printer from the printcap database:

```
$ RUN SYS$SYSTEM:TCPIP$LPRSETUP

Command  < add delete view help exit >: delete
Deleting a printer entry, type '?' for help.

Enter printer name to delete (or view to view printcap file): LOCAL1

                   Symbol  type  value
                   ------  ----  -----
Error log file    :  lf     STR   /SYS$SPECIFIC/TCPIP$LPD/LOCAL1.LOG
Printer Queue     :  lp     STR   LOCAL1
Spool Directory   :  sd     STR   /SYS$SPECIFIC/TCPIP$LPD/LOCAL1

Delete LOCAL1, are you sure? [n]  y

Deleted file: /SYS$SPECIFIC/TCPIP$LPD/LOCAL1.LOG
Deleted files from spooling directory: /SYS$SPECIFIC/TCPIP$LPD/LOCAL1
Removed spooling directory: /SYS$SPECIFIC/TCPIP$LPD/LOCAL1.DIR

Command  < add delete view help exit >: exit
```

22.3.1 Printer Characteristics

You can modify the printer configuration by specifying two-character `printcap` symbols and associated values. Table 22–3 describes the printcap symbols.

Table 22–3 Printcap Symbols

Symbol	Description
sd	Printer spool directory, specified as a UNIX path name.
lf	Error log file, specified as a UNIX path name. This is optional. If you do not specify an error log file, errors are logged to the operator console. An error log can be shared by all local printers if you specify the same file in each printcap printer entry.
lp	Name of the local printer.
ps	LPD PrintServer extensions flag.
rm	Name of the remote host.
rp	Name of the remote printer. The printer name is case sensitive. If you are configuring an LPD print queue to print ASCII text files to an HP LaserJet printer with a JetDirect network card, set the value of the `rp` printcap field to `text`. For example: `:rp = text\` To configure this type of printer for printing PostScript or binary files, set this field to `raw`.
fm	Printer form field. This is equivalent to the OpenVMS command PRINT/FORM. For example, `:fm=CENTER:\` allows the job to print as if the following command were entered: `$ PRINT` *file-name*`/FORM=CENTER` Forms have attributes like print image width and length, or paper stock, which are associated with the print queue when it starts up. To see which forms have been defined for your system, use the DCL command SHOW QUEUE/FORM. To see which form is currently the default for the print queue, enter SHOW QUEUE/FULL.
pa	/PASSALL flag. Tells the print symbiont to ignore any formatting and to send the file to the printer with its format suppressed.

(continued on next page)

Table 22–3 (Cont.) Printcap Symbols

Symbol	Description
nd	/NODELETE flag. Specifies that the temporary file created in TCPIP$LPD for an inbound print job will not be deleted after printing. By default, these temporary files are deleted after printing.
cr	Not supported by TCP/IP Services.
p1-p8	Equivalent to the PRINT/PARAMETER qualifier on the DCL command line. You can specify up to eight optional parameters that are unique to the print symbiont. If the DECprint Supervisor software is running on the system, enter HELP PRINT_PARAMETER for information about the available parameters.

To make the printcap entries easier to read, use one symbol per line, placing a colon (:) at the start of each line and a colon and backslash (:\) at the end of the line to separate the symbols. The last printcap entry ends with a colon (:).

The following sample is an entry from the printcap database that identifies a local printer.

```
#
LOCAL1|local1:\
:lf=/SYS$SPECIFIC/TCPIP$LPD/LOCAL1.LOG:\
:lp=LOCAL1:\
:sd=/SYS$SPECIFIC/TCPIP$LPD/LOCAL1:\
:nd:
```

The following sample is a printcap entry that identifies a remote printer:

```
#
REMOTE1|remote1:\
:lf=/SYS$SPECIFIC/TCPIP$LPD/REMOTE1.LOG :\
:rp=REMOTE1 :\
:rm=hermes :\
:sd=/SYS$SPECIFIC/TCPIP$LPD/ :
```

22.3.1.1 Setting Up Print Spool Directories

Each printer must have its own spool directory located under the SYS$SPECIFIC:[TCPIP$LPD] directory. The spool directory acts as a printer's spooling queue; it contains the files that are queued for printing on that particular printer. A printer spool directory should have the same name as the printer reference name and must be located on the machine to which the printer is attached. Specify the directory using a UNIX-style path name.

Each printer should specify a spool directory even if the printer is connected to another machine or is on another network. You specify a spooling directory in the printcap database with the `sd` symbol. For example:

```
:sd=/SYS$SPECIFIC/TCPIP$LPD/LOCAL1:\
```

22.3.1.2 Setting Up Error Logging

The LPD records printer errors in a log file located in the SYS$SPECIFIC:[TCPIP$LPD] directory. You can set up a separate log file for each printer, or you can set up one to be shared by all local printers.

To specify the log file in the printcap database, use the symbol `lf` and specify the directory as a UNIX path. For example, to specify a log file for the print queue named LOCAL1, the printcap entry would be as follows:

```
:lf=/SYS$SPECIFIC/TCPIP$LPD/LOCAL1.LOG:\
```

To specify a log file that can be shared by all printers, specify the same file for each printer entry. For example:

```
:lp=LOCAL1:\
:lf=/SYS$SPECIFIC/TCPIP$LPD/TCPIP$LPD_LOGFILE.LOG:\

.
.
.

:lp=LOCAL2"\
:lf=/SYS$SPECIFIC/TCPIP$LPD/TCPIP$LPD_LOGFILE.LOG:
```

22.3.1.3 Support for PrintServer Extensions

You can configure LPD to support remote printing on a system that does not implement the PrintServer extensions. You do this for individual queues by adding a ps field in the queue's printcap entry with a value of non_PS. The printcap entry looks as follows:

```
:rm=Remote1
:ps=non_PS
```

If you do not define a ps entry, LPD assumes the printer supports the PrintServer extensions.

Note that you can also configure this option systemwide with the TCPIP$LPD_PS_EXT logical name. Values for this logical name are non_PS and LPS. See Table 22–1 for more information about the LPD logical names.

If a printcap entry does not have a ps field defined, LPD uses the value you assigned with the logical name. If the logical name is not defined, LPD uses PrintServer extensions as the default.

22.4 Managing LPD Server Queues

To start the LPD server queues, enter the following command:

```
$ @SYS$STARTUP:TCPIP$LPD_STARTUP
```

To stop the LPD server queues, enter the following command:

```
$ @SYS$STARTUP:TCPIP$LPD_SHUTDOWN
```

To display the status of a remote queue, enter the LPQ command at the DCL prompt. To remove jobs from a remote printer queue, enter the LPRM command at the DCL prompt. See *DIGITAL TCP/IP Services for OpenVMS User's Guide* for more information about these commands.

The following example deletes all the jobs on remote print queue EIDER_DOWN_Q:

```
$ LPRM EIDER_DOWN_Q /ALL
```

22.5 Controlling Access to Local Queues

You can grant or deny remote users access to the LPD server by entering the command SET SERVICE LPD /FLAGS=APPLICATION_PROXY. This causes LPD to authenticate remote users through the TCP/IP Services proxy database. You identify the remote users by adding communication proxy entries in the proxy database, TCPIP$PROXY.DAT. Each remote user allowed to access your local queues must have an entry.

To add a proxy entry, enter:

```
TCPIP> ADD PROXY user_name /HOST=host_name /REMOTE_USER=user_name
```

For each host, define both its host name and alias name. If you need to use lowercase letters to specify a remote user name, enclose it in quotation marks. For example:

```
/REMOTE_USER="unixuser"
```

You use wildcard characters when adding proxy entries for users on remote systems. For example, the following command allows any user on the remote host REMOTE1 to submit print jobs to the print queues on your system.

```
TCPIP> ADD PROXY R_USERS /HOST=REMOTE1 /REMOTE_USER="*"
```

To disable authentication, use the /FLAG=NOAPPLICATION_PROXY option to the SET SERVICE LPD command. Use the /REJECT option to deny access from certain hosts. For example:

```
TCPIP> SET SERVICE LPD /REJECT=HOSTS=(loon,ibis,tern)
```

22.6 Receiving LPR/LPD OPCOM Messages

The LPR/LPD spooler can notify you of selected events with OPCOM messages. To receive these notifications, enter:

```
$ TCPIP SET SERVICE LPD /LOG=option
$ REPLY /ENABLE=OPCOM
```

The logging options are:

- LOGIN — LPD receiver startup and exit
- LOGOUT — Job completion
- ACTIVATE — Queue startup

22.7 Using OpenVMS Flag Page Options

LPD supports all OpenVMS flag page print options, including:

- /FLAG qualifier of the DCL PRINT command
- /DEFAULT=FLAG setting on the LPD print queue
- /SEPARATE=FLAG setting on the LPD print queue

To enable these features, define the system logical name TCPIP$LPD_VMS_FLAGPAGES. This logical name applies to all print queues:

```
$ DEFINE /SYSTEM TCPIP$LPD_VMS_FLAGPAGES
```

When you define TCPIP$LPD_VMS_FLAGPAGES, LPD does the following:

- Obeys the OpenVMS instructions regarding flag pages for outbound jobs.
- Submits inbound jobs with /FLAG or /NOFLAG, based on the presence of the L card directive in the LPD control file received from the remote host.

  Inbound jobs with an L card directive are submitted to the destination print queue as PRINT /FLAG.

  Inbound jobs without an L card directive are submitted to the destination print queue as PRINT /NOFLAG.

- Renders meaningless the /PARAMETERS=NOFLAG qualifer to the DCL command PRINT.

22.8 Solving LPD Problems

In addition to the log files specified in the printcap database, which is used by the LPR and LPD symbionts, the LPD receiver logs diagnostic messages to the log file TCPIP$LPD_RCV_STARTUP.LOG. Use the TCPIP$LPD_RCV and TCPIP$LPD_DEBUG logical names to control LPR/LPD diagnostic information in these logs.

Table 22–1 describes the logical names in more detail.

If you have problems, turn on all the LPR/LPD diagnostics. That is, define TCPIP$LPD_DEBUG and TCPIP$LPD_RCV as 15. Leaving these diagnostics on during normal use might affect the performance of LPD and produce large log files.

23

Setting Up and Managing TELNETSYM

The TELNET print symbiont (TELNETSYM) provides remote printing services that enable the use of standard OpenVMS printing features not available with the LPR/LPD print service. With TELNETSYM configured on your system, you can set up and manage a remote printer attached to a remote terminal server as if it were directly connected to your system. The TELNET symbiont functions in a manner that is similar to that of LATSYM for Compaq's local area transport (LAT) software.

The TELNET symbiont performs the following functions:

- Transfers record-oriented data to printers.
- Configures printers attached to terminal servers that support TELNET.
- Supports outbound print jobs and offers standard OpenVMS preformatting for outbound print jobs.

This chapter reviews key TELNETSYM concepts and describes:

- How to start up and shut down the TELNETSYM print service (Section 23.2)
- How to set up a TELNETSYM print queue (Section 23.3)
- How to set up a relay queue (Section 23.4)
- How to managing and customize TELNETSYM print queues (Section 23.5)
- How to solve TELNETSYM problems (Section 23.6)

23.1 Key Concepts

TELNETSYM is a true OpenVMS print symbiont; it performs all print formatting functions, such as header and trailer page generation, pagination, queuing, and handling of multiple forms. TELNETSYM extends the OpenVMS print symbiont by redirecting its output to a network (TELNET) channel.

TELNETSYM sets its process names to TCPIP$TNSYM1, TCPIP$TNSYM2, and so on. Each TELNETSYM process can control up to 16 print queues. You can control the maximum number of print queues by defining the TCPIP$TELNETSYM_STREAMS logical, as described in Section 23.5.6.

23.1.1 TELNETSYM Modifications to the Output Stream

TELNETSYM adds escape (0xFF) bytes in the data stream so they are not mistakenly interpreted as TELNET protocol IAC commands.

TELNETSYM doubles any TELNET IAC characters found in the byte stream unless TCPIP$TELNETSYM_RAW_TCP is defined for the queue. The IAC character is a hexadecimal FF.

If the print job is queued with the /PASSALL qualifier, TELNETSYM sets up a binary TELNET channel by inserting IAC-DO-BINARY and IAC-WILL-BINARY escape sequences.

You can turn off this behavior by defining the logical name TCPIP$TELNETSYM_RAW_TCP for the queue. If you set this logical name, none of this processing is done.

The IAC-DO-BINARY sequence is 6 bytes, which are symbolically:

IAC, DO, BINARY, IAC, WILL, BINARY

The hexadecimal equivalents are:

```
FF,FD,00,FF,FB,00
```

TELNETSYM does not add any additional data to the stream other than that described. It does not insert form feed characters that were not present in the output from the OpenVMS print symbiont. Therefore, any additional characters observed as added to a print job come from the OpenVMS or other print symbiont (for example, Compaq *PATHWORKS*/Advanced Server for OpenVMS).

TELNETSYM can remove (suppress) any form feed (0x0c) characters that the OpenVMS print symbiont adds to the beginning or end of print jobs. Use the TCPIP$TELNETSYM_SUPPRESS_FORMFEEDS logical name to control this function, as described in Section 23.6.4.1.

23.2 TELNETSYM Service Startup and Shutdown

The TELNETSYM service can be shut down and started independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$TELNETSYM_STARTUP.COM allows you to start up the TELNETSYM service independently.
- SYS$STARTUP:TCPIP$TELNETSYM_SHUTDOWN.COM allows you to shut down the TELNETSYM service independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$TELNETSYM_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the TELNETSYM service is started.
- SYS$STARTUP:TCPIP$TELNETSYM_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the TELNETSYM service is shut down.

23.3 Setting Up Print Queues

Use the DCL command INITIALIZE/QUEUE to set up a TELNETSYM queue. Use the /PROCESSOR and /ON qualifiers as follows:

1. Specify the TELNETSYM image name in the /PROCESSOR qualifier, as follows:

   ```
   /PROCESSOR=TCPIP$TELNETSYM
   ```

2. Specify the host name and port number to which the queue sends the print data with the /ON qualifier, as follows:

   /ON="*hostname:portnumber*"

For example, to set up a TELNETSYM queue named xyz_q to print using TELNETSYM to host printserver.xyz.com at TCP port 4242, enter:

```
$ INITIALIZE /QUEUE /PROCESSOR=TCPIP$TELNETSYM -
_$ /ON="printserver.xyz.com:4242" xyz_q
```

23.4 Setting Up Relay Queues

You can redirect the output of TELNETSYM to another queue rather than sending it directly to a remote printer. A queue with this setup is a **relay queue**. Use relay queues to funnel fully formatted output to an outbound LPD queue. LPD transfers jobs that are fully formatted on the sending side by OpenVMS.

In this case, TELNETSYM saves the output stream to a temporary file and then submits the file to the destination queue. TCP/IP Services is not used.

To set up a TELNETSYM relay queue, specify the /ON qualifier of the INITIALIZE/QUEUE command as follows, where *qname* is the name of the queue to which you want TELNETSYM to send its output.

/ON="TCPIP$QUEUE:*qname*"

To set up a TELNETSYM relay queue named RELAYQ_4 to send output to the queue named LPD_Q4, enter:

```
$ INITIALIZE /QUEUE /ON="TCPIP$QUEUE:LPD_Q4" -
_$ /PROCESS=TCPIP$TELNETSYM /DEVICE=PRINTER RELAYQ_4
```

23.5 Managing and Customizing Your Print Queues

You can manage and customize TELNETSYM for each print queue by defining logical names before you start the queue. Because the logical names are translated once at queue startup time, they can be defined differently for each TELNETSYM queue. Use the /SYSTEM qualifier when defining TELNETSYM logical names. You must stop and restart the print queue to establish the changes you make with logical names.

Some TELNETSYM configuration logical names are used to set a configuration option either ON or OFF. If the logical name is defined, the option is ON. If it is not defined, the option is OFF.

Other logical names require a specific value. The following sections describe TELNETSYM logical names. The descriptions indicate when a value is required.

23.5.1 Controlling Stream of Print Bytes Sent Over the Link

If a remote printer supports a raw network data connection rather than the TELNET protocol, you can print to such a printer by suppressing all TELNET modifications of the output stream with the following logical names:

- TCPIP$TELNETSYM_RAW_TCP

  Suppresses all TELNET type modifications of the print output stream.

  This logical name also prevents the TELNETSYM from doubling IAC characters and sending the TELNET escape sequence to negotiate binary options for files printed /PASSALL.

- TCPIP$TELNETSYM_SUPPRESS_FORMFEEDS

  Suppresses form feeds between jobs. This includes the form feed that is normally sent before the first job printed to a print queue and the form feed sent at the end of every job. For more information, see Section 23.6.4.1.

23.5.2 Setting Up Error Logging

OPCOM messages sent by TELNETSYM include the name of the execution queue. In addition, each TELNETSYM queue has a log file named TCPIP$TELNETSYM_*queue-name*.LOG.

By default, TELNETSYM sends messages to the operator and records error and informational messages in the file TCPIP$TELNETSYM_*queue-name*.LOG. This file is located in SYS$SPECIFIC:[TCPIP$LPD].

You can use logical names to modify the way the TELNETSYM logs information and the type of information it reports. For example, TELNETSYM can log diagnostic messages that you can use when troubleshooting problems with a link.

Use the following logical names to modify error logging:

- TCPIP$TELNETSYM_VERBOSE

  Turns on the logging of TELNETSYM diagnostics to the file TCPIP$TELNETSYM.LOG. These diagnostics include informational messages that indicate when links have come up or gone down and error messages.

- TCPIP$TELNETSYM_NO_OPCOM

  Stops TELNETSYM from sending messages to the operator console.

- TCPIP$TELNETSYM_DEBUG

  Used with TCPIP$TELNETSYM_VERBOSE, this logical name tells TELNETSYM which diagnostic message types to log.

  Specify a value for each bit. Each bit set in the value turns on a particular logging function. The options are:

Bit 0	Tracks the flow of code. For example: `xyz-n-xyz-routine entered`
Bit 1	Tracks the allocation of memory. For example: `just freed address 7F0000`
Bit 2	Logs the bytes sent and received over TCP/IP link.

  To set a bit, assign the value to the logical name whose binary equivalent would have the bit set. For example, you can tell TELNETSYM to log everything that it writes to and receives from the TCP/IP link by entering:

  ```
  $ DEFINE /SYSTEM TCPIP$TELNETSYM_DEBUG 4
  ```

  Decimal 4 is binary 100 with bit 2 set. Note that you can achieve different combinations by setting more than one bit in the value. A value of 3, for example, sets bits 0 and 1, causing logging of flow of code and memory allocation diagnostics.

  If TCPIP$TELNETSYM_DEBUG is undefined, TELNETSYM does not log these diagnostics.

Bit 2 is useful in unassisted problem solving. Be aware, however, that the log file can become large because all the data sent over the link to the printer is logged. Bits 0 and 1 are primarily for use by Compaq. However, with knowledge of PSM$ symbionts, you might find all the options useful.

- TCPIP$TELNETSYM_LOG_KEEP

  By default, TELNETSYSM saves all log files. Define this logical name to limit the number of log files saved. The value assigned to this logical name is the number of versions of a log file TELNETSYM will allow before it starts purging. When the number of files reaches the number you specified, TELNETSYM starts purging files.

  For example, to configure the queue to purge after more than three copies of the same log file are created, define the logical name as follows:

  ```
  $ DEFINE /SYSTEM TCPIP$TELNETSYM_LOG_KEEP 3
  ```

- TCPIP$TELNETSYM_SCRATCH

  By default, TELNETSYM stores log files and any temporary files created by relay queues in the directory SYS$SPECIFIC:[TCPIP$LPD]. You can change the default directory for one or all of your TELNETSYM queues. For example:

  ```
  $ DEFINE /SYSTEM TCPIP$TELNETSYSM_SCRATCH device:[directory.path]
  ```

  If you define the logical name TCPIP$TELNETSYM_SCRATCH, the log files are stored in the TCPIP$TELNETSYM_SCRATCH directory.

  If you do not define TCPIP$TELNETSYM_SCRATCH, the log files are stored in TCPIP$LPD_SPOOL.

  If TCPIP$LPD_SPOOL is not defined, the log files go into the SYS$SPECIFIC:[SYSEXE] directory.

23.5.3 Controlling Characteristics of the TCP/IP Link

The TELNETSYM configuration logical names allow you to set TELNETSYM parameters. To see the default values for these parameters, enter the following command:

```
TCPIP> SHOW PROTOCOL TCP /PARAMETER
TCP
  Delay ACK:              enabled
  Window scale:           enabled
  Drop count:                   8
  Probe timer:                150

                          Receive                Send

  Push:                  disabled            disabled
  Quota:                    61440               61440
```

The logicals that you can use to modify these parameters are:

- TCPIP$TELNETSYM_KEEPALIVE

  Controls KEEPALIVE processing.

  The KEEPALIVE timer is used to periodically test the other end of a link that appears to be idle. Its purpose is to detect when a remote host has failed or has been brought down, or when the logical connection has been broken.

  For TELNETSYM, you control this timer with the following command:

  ```
  $ DEFINE/SYS TCPIP$TELNETSYM_KEEPALIVE 1
  ```

This logical name is not used by the server; it is used by the TELNET client. If you are changing this logical name, then there is no need to restart TCP/IP Services. If this logical is defined, the KEEPALIVE function is enabled.

By default, the KEEPALIVE timer is disabled. Broken connections will be detected only when the relevant application sends data.

- TCPIP$TELNETSYM_DROPTIME

  The DROP timer indicates how long a connection should be maintained (after repeated timeouts) before closing the connection. The DROP timer is in effect only after the link has been established, and it takes effect only if the KEEPALIVE logical is also defined.

  You control the DROP timer using the following command:

  ```
  $ DEFINE/SYS TCPIP$TELNETSYM_DROP x
  ```

  where *x* specifies the number of seconds to hold the connection before closing it.

  The default value for the DROP timer is 300 seconds.

  Note that the value for the DROP timer must be greater than the value for the PROBE timer. When you define only one of these TELNETSYM logical names, the default value will be used for the other logical name.

- TCPIP$TELNETSYM_PROBETIME

  This logical name allows you to control the PROBE timer.

  The PROBE timer defines:

  – When establishing an initial connection, the number of seconds TCP/IP Services will wait for a response before a timeout occurs. You can enable this function even if the KEEPALIVE timer is disabled.

  – The length of time allowed to pass before TCP/IP Services checks an idle connection. This function requires that the KEEPALIVE timer be enabled.

  You control the PROBE timer using the following command:

  ```
  $ DEFINE/SYS TCPIP$TELNETSYM_PROBE x
  ```

  In this command, *x* specifies the number of seconds to wait before timing out the connection.

  The value of the PROBE timer must always be less than or equal to the value of the DROP timer. The default value for the PROBE timer is 75 seconds.

- TCPIP$TELNETSYM_SNDBUF

  Specifies the size of the socket send buffer that TELNETSYM uses.

23.5.4 Establishing a TELNETSYM Link

If a network link has not been established, the TELNET symbiont attempts to establish one. Printing starts when the link is successfully established. The TELNET symbiont continues to try to establish a network link until it is successful or until a retry interval you define has expired.

The logical name TCPIP$TELNETSYM_RETRY_INTERVAL defines the time for TELNETSYM to wait between link-establishment retries when link establishment has failed. The value for this logical name is an OpenVMS delta time.

If this logical name is not defined, TELNETSYM defaults to a wait period of 3 minutes between retries.

For example, to define a retry interval of 30 seconds, enter:

```
$ DEFINE /SYSTEM TCPIP$TELNETSYM_RETRY_INTERVAL "0 00:00:30.00"
```

23.5.5 Releasing a TELNETSYM Link

By default, TELNETSYM releases an established link at the end of a print job. This behavior is useful when multiple systems contend for the same printer. Configuring TELNETSYM to release the link at the end of a job allows other systems to print quickly. However, this behavior can also be a disadvantage because of the overhead involved with link creation for each print job.

When there is little or no contention for a printer, it is useful to configure TELNETSYM to release the link only after a certain period of idle time has passed. With this approach, TELNETSYM waits for the configured idle time to elapse and then closes the link. This option works well within batch printing applications.

Use the logical name TCPIP$TELNETSYM_IDLE_TIMEOUT to define the length of time to wait before terminating an inactive link. Specify a value that is an OpenVMS delta time.

For example, to define a link-idle-timeout of 10 minutes, enter:

```
$ DEFINE /SYSTEM TCPIP$TELNETSYM_IDLE_TIMEOUT "0 00:10:00.00"
```

Idle time occurs during printing as well as between print jobs. Any idle time on the link can cause a timeout. Therefore, it is important to adjust the value of this logical carefully.

23.5.6 Setting the Number of Execution Queues

The logical name TCPIP$TELNETSYM_STREAMS defines the number of execution queues handled by each TELNETSYM process. The value you enter (a number from 1 through 16) when defining this logical name is passed to the PSM$PRINT system routine. The default is a maximum of 16 queues per symbiont process.

Use this logical to turn TELNETSYM into a single-threaded symbiont (*value*=1) in which each queue runs its own process. This makes diagnosing problems easier and lessens the consequences of a failure.

If you are defining this logical name, define it once. Do not define it differently for each TELNETSYM print queue.

23.6 Solving TELNETSYM Problems

To avoid potential problems with TELNET printing, be aware of the following guidelines and considerations.

23.6.1 Using TCPIP$TELNETSYM for the First Time

If you use the public domain TELNET symbiont and want to switch to the TCP/IP Services TELNET symbiont, you must change the value of the /PROCESSOR qualifier on the TELNET symbiont queues. When you do this, include any command procedures that start up the queues. Change:

```
/PROCESSOR=TELNETSYM
```

to:

```
/PROCESSOR=TCPIP$TELNETSYM
```

23.6.2 Printing to Terminal Servers

When you print to a terminal server system, ensure that:

- Input flow control is disabled for the port to which you are printing.

  Enter:

  ```
  > CHANGE PORT port INPUT FLOW DISABLED
  ```

- The TELNET server for the terminal server port is set to recognize a new line as a carriage-return character followed by a line feed character.

  Enter:

  ```
  > CHANGE PORT port TELNET SERVER NEWLINE TO HOST
  ```

23.6.3 Stalled Print Queues

When you print a job to a TELNETSYM queue, a link must be established between the queue and the printer. If there is high contention for the printer, it might be busy, causing the first attempt to fail.

TELNETSYM continues to try to establish the link, according to the retry interval logical name TCPIP$TELNETSYM_RETRY_INTERVAL. Until the link is established, the execution queue stalls. When the link comes up, the job prints. A stalled TELNETSYM queue is not necessarily an error.

If the queue stalls while printing a job, the printer probably requires human intervention; that is, the printer is out of paper or jammed.

If TELNETSYM causes a print queue to fail, reset the queue. Enter the following command:

```
$ STOP /QUEUE /RESET queue-name
```

23.6.4 Solving Formatting Problems

To track down problems with improper formatting on the printed page (for example, "garbage" for a graphics file or unwanted blank pages), use bit 2 of the TELNETSYM logical name TCPIP$TELNETSYM_DEBUG. Defining this logical helps determine whether the source of the problem is TELNETSYM. Follow these steps:

1. Define the logical as 4 in the system table. Enter:

   ```
   $ DEFINE /SYSTEM TCPIP$TELNETSYM_DEBUG 4
   $ STOP /QUEUE /RESET TELNETSYM_queue-name
   $ START /QUEUE TELNETSYM_queue-name
   ```

2. Print the job that does not print properly.

3. Look at the TELNETSYM log file for the queue.

   This file has messages that show you every byte sent over the link to the printer, such as control characters and setup/reset modules.

   If the raw TCP logical name is not defined, you will see doubled IAC characters (hexadecimal FF).

   If you print /PASSALL with the raw TCP logical name not defined, the job starts with the TELNET options negotiation sequence "do binary, will binary."

4. Identify the problem. Either fix it or report it to your Compaq support representative. Keep in mind that the OpenVMS print symbiont may be the cause of the problem. TELNETSYM only modifies the output as described in Section 23.1.1.
5. Turn off debug mode.
6. Start the TELNETSYM queue.

23.6.4.1 Controlling Form Feed Suppression

Use the TCPIP$TELNETSYM_SUPPRESS_FORMFEEDS logical to control the suppression of form feeds. The bit settings you specify in the value control the time of the operation and the type of form feed suppression to perform:

- Bits 0 and 1 specify when to do form feed suppression. It can be done at either job setup or job completion time, or both. At least one of these bits must be set to enable form feed suppression.
- Bits 4 and 5 together specify how to perform form feed suppression. With TCP/IP Services, you can set either of two levels of form feed suppression. Both levels eliminate the form feed character from the stream of output bytes that is sent when the queue is first started.
  - Level 1 form feed suppression operates similarly to form feed suppression under previous versions of TCP/IP Services. It will not eliminate subsequent form feed characters, but will instead substitute a line feed character for the form feed character. As a result, what would have been a carriage-return/form feed sequence in the output stream becomes a carriage-return/linefeed sequence.
  - Level 2 form feed suppression eliminates all form feed characters and carriage-return/form feed sequences from the output stream.

The following examples show how to calculate the value for the logical name:

1. This example shows how to determine the value of the TCPIP$TELNETSYM_ SUPPRESS_FORM FEEDS logical if you want level 2 form feed suppression at both job setup and job completion times. The value of the logical is determined by the following bit settings:

```
63 <-----  unused -----> 6|54|32|1|0
_____________________________________
| XXXXXXXXXXXXXXXXXXXXXXXX|ll|XX|s|c|
-------------------------------------
                           10 00 1 1    --->---> binary 100011
                           ^^    ^ ^
                            |    | |
10 is binary for decimal ----+    +-+--<-- set both the setup and comp. bits
2 is level 2
```

   The binary value for level 2 form feed suppression at both job setup and job completion time is 100011 (hexadecimal 23 or decimal 35). Because the value of the logical is a decimal value, you define it as follows:

```
$ DEFINE/SYSTEM TCPIP$TELNETSYM_SUPPRESS_FORM FEEDS 35
```

2. This example shows how to determine the value of the TCPIP$TELNETSYM_SUPPRESS_FORM FEEDS logical if you want level 1 form feed suppression at job completion time only. The value of the logical is determined by the following bit settings:

```
63 <-----  unused -----> 6|54|32|1|0
_____________________________________
| XXXXXXXXXXXXXXXXXXXXXXX|ll|XX|s|c|
-------------------------------------
                           01 00 0 1    --->---> binary 010001
                           ^^    ^ ^
                            |    | |
01 is binary for decimal ----+    +-+--<-- set only the comp. bit
1 is level 1
```

The binary value for level 1 form feed suppression at job completion time only is 010001 (hexadecimal 11 or decimal 17). Because the value of the logical is a decimal value, you define it as follows:

```
$ DEFINE/SYSTEM TCPIP$TELNETSYM_SUPPRESS_FORM FEEDS 17
```

23.6.4.2 Buffer Dumps

TELNETSYM logs control characters and nonprinting characters by preceding the hexadecimal value of the byte with a backslash. For example, the following sequence:

```
Carriage Control
Form Feed
Carriage Control
Line Feed
Tab
the text "Use Your Screen Saver to Conserve Energy."
Carriage Return
Line Feed
```

is logged as:

```
\0D\0C\0D\0A\09Use Your Screen Saver to Conserve Energy.\0D\0A
```

The "do binary, will binary" sequence starting off a /PASSALL job appears as:

```
\FF\FD\00\FF\FB\00
```

24
Setting Up PC-NFS

The PC-NFS server provides authentication and print services for personal computers running PC-NFS. Users on a PC client can associate the name of the PC printer with an OpenVMS print queue and print files to the associated queue. To access the PC-NFS server, PC users must have an entry in the proxy database and have corresponding OpenVMS accounts on the server.

This chapter describes:

- How to start up and shut down the PC-NFS server (Section 24.1)
- How to provide PC-NFS printing services (Section 24.2)
- How to manage PC-NFS print queues (Section 24.3)
- PC-NFS authentication (Section 24.4)

For information about setting up NFS proxy identities for PC-NFS client users, see Chapter 20.

24.1 PC-NFS Startup and Shutdown

The PC-NFS server can be shut down and started up independently of TCP/IP Services. This is useful when you change parameters or logical names that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$PCNFS_STARTUP.COM allows you to start up the PC-NFS server independently.
- SYS$STARTUP:TCPIP$PCNFS_SHUTDOWN.COM allows you to shut down the PC-NFS server independently.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$PCNFS_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the PC-NFS server is started.
- SYS$STARTUP:TCPIP$PCNFS_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the PC-NFS server is shut down.

24.2 Providing PC-NFS Print Services

To configure PC-NFS print services, you must create and export a spool directory and define two system logical names. Follow these steps when configuring your print server for printing by PC-NFS clients:

1. If one does not already exist, create a spool directory.
2. Map the OpenVMS device to the spool directory path name. For example:

   ```
   TCPIP> MAP "/PC_PRINT/WORK"  DSA31:
   ```

3. Make the path available with the ADD EXPORT command as follows:

   ```
   TCPIP> ADD EXPORT "/PC_PRINT/WORK" /HOST=* /OPTIONS=TYPELESS_DIRECTORIES
   ```

4. Create or edit the SYS$STARTUP:TCPIP$PCNFS_SYSTARTUP.COM file to include the following logical name definitions:

   ```
   DEFINE /SYSTEM TCPIP$PCNFSD_SPOOLDEV DSA31:
   DEFINE /SYSTEM TCPIP$PCNFSD_SPOOLEXPORT "/PC_PRINT/WORK"
   ```

   The logical name TCPIP$PCNFSD_SPOOLDEV specifies the device name for the spool device; TCPIP$PCNFSD_SPOOLEXPORT specifies the exported spool directory.

24.3 Managing PC-NFS Print Queues

PC users can associate the name of the DOS printer you are configuring with an OpenVMS print queue and print files to the associated queue. PC clients cannot, however, manage NFS print queues from their PC. To manage print queues, you must log in to either a privileged account or the PC's proxy account on the NFS server host, and enter DCL commands to:

- List jobs queued from the PC
- Cancel queued jobs
- Obtain a list of available print queues
- Obtain status of a particular print queue

24.4 PC-NFS Authentication

When accessing files on an NFS server, a PC user obtains authentication once from any host running PC-NFS. The user can also access NFS files on that host or other hosts, even if the user's UID/GID has proxy mappings to a different OpenVMS account on each TCP/IP host.

However, with PC-NFS printing, if the PC user obtains authentication from one host, the user can only print successfully on other TCP/IP Services hosts that have a valid OpenVMS account for the same user name.

Part 7
Appendixes

Part 7 contains the following appendixes:

- Appendix A, Gateway Routing Daemon (GATED) Configuration Reference, describes how to configure GATED protocols for use with the Gateway Routing Daemon (GATED).
- Appendix B, EBCDIC/DMCS Translation Tables, provides EBCDIC/DMCS translation tables.
- Appendix C, How NFS Converts File Names, describes how NFS converts UNIX file names to OpenVMS file names.
- Appendix D, Acronyms, contains a list of acronyms for OpenVMS and networking.

A

Gateway Routing Daemon (GATED) Configuration Reference

This appendix describes how to configure the Gateway Routing Daemon (GATED).

A.1 The GATED Configuration File

You must configure the GATED protocols before starting GATED routing by editing the configuration file TCPIP$GATED.CONF, located in SYS$SYSDEVICE:[TCPIP$GATED]. A template file TCPIP$GATED.TEMPLATE is also available in this directory.

The file TCPIP$GATED.CONF contains statements that select routing protocols, manage routing information, manage independent system routing and control tracing options.

After editing the configuration, enter the TCP/IP management command TCPIP START ROUTING/GATED to start the GATED process. If the configuration file is not formatted correctly, GATED will not be able to parse the file and GATED will terminate.

If you make changes to the GATED configuration file after the GATED process is already running, you must stop GATED by entering the command TCPIP STOP ROUTING/GATED. Then restart the GATED process to make the changes take affect.

See *Compaq TCP/IP Services for OpenVMS Management Command Reference* for detailed descriptions of the SET GATED and START ROUTING/GATED commands.

A.2 Configuration File Statement Syntax

Parameters shown in brackets ([ ]) show optional keywords and parameters. The vertical bar (|) indicates a choice of optional parameters. Parentheses (( )) group keywords and parameters, when necessary. For example:

```
[backbone | (area area)]
```

In this example, the brackets indicate that either parameter is optional. The keywords are `backbone` and `area`. The vertical bar indicates that either `backbone` or `area` *area* can be specified. Because *area* is in italics, it is a parameter that you provide.

The following comment styles are valid in a GATED configuration file. (Comments may appear anywhere in the file.)

- A pound sign (#)

- The C-style comments that start with /* and end with */

Note

In a GATED configuration file, statements end with a semicolon (;). Do not use a semicolon as a comment character in your configuration file. Anything following a semicolon is interpreted as the start of the next statement.

A.3 Statement Grouping

The configuration file consists of statements grouped in the following order:

1. Options statements
2. Interface statements
3. Definition statements
4. Protocol statements
5. Static statements
6. Control statements
7. Aggregate statements

Note

Entering a statement out of order causes an error when parsing the configuration file.

The following statements do not fit in the above categories:

- `%directive` statements
- `%trace` statements

These statements provide instructions to the parser, and control tracing from the configuration file. They do not control the configuration of any protocol and may occur anywhere in the configuration file.

A.4 Configuration Statements

Table A–1 describes each TCPIP$GATED.CONF configuration statement.

Table A–1 GATED Configuration Statements

Command	Type	Description
`%directory`	directive	Sets the directory for `include` files.
`%include`	directive	Includes a file into TCPIP$GATED.CONF.
`traceoptions`	trace	Specifies which events are traced.

(continued on next page)

Table A–1 (Cont.) GATED Configuration Statements

Command	Type	Description
options	definition	Defines GATED options.
interfaces	definition	Defines GATED interfaces.
autonomoussystem	definition	Defines the autonomous system (AS) number.
routerid	definition	Defines the originating router (BGP, OSPF).
martians	definition	Defines invalid destination addresses.
rip	protocol	Enables the RIP protocol.
kernel	protocol	Configures kernel interface options.
ospf	protocol	Enables the OSPF protocol.
egp	protocol	Enables the EGP protocol.
bgp	protocol	Enables the BGP protocol.
redirect	protocol	Configures the processing of ICMP redirects.
icmp	protocol	Configures the processing of general ICMP packets.
snmp	protocol	Enables reporting to SNMP.
static	static	Defines static routes.
import	control	Defines which routes to import.
export	control	Defines which routes to export.
aggregate	control	Defines which routes to aggregate.
generate	control	Defines which routes to generate.

A.5 Creating the GATED Configuration File

To create a configuration file for your local host, edit a copy of the sample file TCPIP$GATED.TEMPLATE (located in the SYS$SYSDEVICE:[TCPIP$GATED] directory),then save the file to SYS$SYSDEVICE:TCPIP$GATED.CONF.

The following shows the template configuration file:

```
#------------------------------------------------------------------------------
#
# TCPIP$GATED.CONF - Sample config file, preconfigured for RIP v1.
#
#------------------------------------------------------------------------------
#
interfaces {
 interface all passive ;
 };
#
# Protocols:
#
rip on {
 broadcast;
 interface all ripin ripout version 1;
 };
#
redirect on;
routerdiscovery server off;
hello off;
ospf off;
egp off;
bgp off;
snmp off;

#
# Static routes:
#
#static {
# 10.1.2.0 mask 255.255.255.0 gateway 10.1.1.1;
# default gateway 10.1.2.3;
# };
#
# Policy:
#
#export proto rip {
# proto static { all metric 1; };
# proto direct { all; };
# proto rip { all; };
# };
```

A.6 Defining Preferences and Routing

The configuration file can define routes from one protocol or peer to another, assigning each route a value, called a **preference**.

The preference value determines the order of routes to the same destination in a single routing database. The active route is chosen by the lowest preference value. Some protocols implement a second preference (preference2), sometimes referred to as a "tie breaker."

Preferences have the following characteristics:

- May appear in several different configuration statements in the configuration file. Be aware, however, that the last, or most specific value set for a route is the value GATED will use.
- May specify one network interface over another, one protocol over another, or one remote gateway over another.
- Cannot be used to control the selection of routes within an interior gateway protocol (IGP). That function is accomplished automatically by the protocol based on metric.

- May select routes from the same exterior gateway protocol (EGP) learned from different peers or autonomous systems.

The GATED daemon selects a route based on the following preference criteria:

- The route with the best (numerically smallest) preference is selected.
- If the two routes have the same preference, the route with the best (numerically smallest) `preference2` is selected.
- A route from an IGP is selected over a route from an EGP. The least preferred is a route learned indirectly by an IGP from an EGP.
- If autonomous system (AS) path information is available, it is used to help determine the most preferred route as follows:
  - A route with an AS path is selected over one without an AS path.
  - If the AS paths and origins are identical, the route with the lower metric is selected.
  - A route with an AS path origin of IGP is preferred over a route with an AS path origin of EGP. The least preferred is an AS path with an unknown origin.
  - A route with a shorter AS path is preferred.
  - If both routes are from the same protocol and AS, the one with the lowest metric is selected.
  - The route with the lowest numeric next-hop address is used.

A.6.1 Assigning Preferences

A default preference is assigned to each source from which GATED receives routes. Preference values range from 0 to 255, with the lowest number indicating the most preferred route.

Table A–2 lists each type of route, the statement (or clause within statements) that sets preference for the route, and the default preference for each type of route.

Note that a statement that is narrow in scope has a higher precedence given to its preference value, but affects a smaller set of routes.

Table A–2 Default Preference Values

Preference	Defined by Statement	Default
Direct connected networks	interface	0
OSPF routes	ospf	10
Internally generated default	gendefault	20
Redirects	redirect	30
Routes learned through route socket	kernel	40
Static routes from config	static	60
ANS SPF (SLSP) routes	slsp	70
HELLO routes	hello	90

(continued on next page)

Table A–2 (Cont.) Default Preference Values

Preference	Defined by Statement	Default
RIP routes	rip	100
Point-to-point interface		110
Routes to interfaces that are down	interfaces	120
Aggregate/generate routes	aggregate/generate	130
OSPF AS external routes	ospf	150
BGP routes	bgp	170
EGP	egp	200

A.6.2 Sample Preference Specifications

In the following example, the preference applicable to routes learned through RIP from gateway 138.66.12.1 is 75. The last preference applicable to routes learned through RIP from gateway 138.66.12.1 is defined in the accept statement. The preference applicable to other RIP routes is found in the rip statement. The preference set on the interface statement applies only to the route to that interface.

```
interfaces {
        interface 138.66.12.2 preference 10 ;
} ;
rip yes {
    preference 90 ;
} ;
import proto rip gateway 138.66.12.1 preference 75 ;
```

A.7 Tracing Options

You can specify tracing options at the following levels: file specifications, control options, and global and protocol specific tracing options. Unless overridden, tracing options from the next higher level are inherited by lower levels. For example, Border Gateway Protocol (BGP) peer tracing options are inherited from BGP group tracing options, which are inherited from global BGP tracing options, which are inherited from global GATED tracing options. At each level, tracing specifications override the inherited options.

The syntax for trace options statements is as follows:

```
traceoptions [trace_file [replace] [size size[k|m]
files files]]
[control_options] trace_options[except trace_options] ;
traceoptions none ;
```

Table A–3 describes the valid trace options.

Table A–3 Trace Options

Option	Definition
trace_file	Specifies the file to receive tracing information. If this file name does not begin with a slash (/), the directory in which GATED was started is prepended to the name.
replace	Replaces an existing trace file. The default is to append to an existing file.
size *size*[k|m] files *files*	Limits the maximum size of the trace file to the specified size (minimum 10 kilobytes). When the trace file reaches the specified size, it is renamed to file.0, then file.1, file.2, up to the maximum number of files (minimum specification is 2).
control_options	Specifies options that control the appearance of tracing. The only valid value is nostamp, which specifies that a timestamp should not be prepended to all trace lines.
except trace_options	Enables a broad class of tracing and then disables more specific options.
none	Specifies that all tracing should be turned off for this protocol or peer.

A.7.1 Global Tracing Options

There are two types of global options: those with global significance (Table A–4) and those with protocol significance (Table A–5).

Table A–4 Global Significance Options

Option	Definition
parse	Traces the lexical analyzer and parser. Used mainly by GATED developers for debugging.
adv	Traces the allocation of and freeing of policy blocks. Used mainly by the GATED developers for debugging.
symbols	Traces symbols read from the kernel at startup. The principal way to specify this level of tracing is by the -t option on the command line, because the symbols are read from the kernel before parsing the configuration file.
iflist	Traces the reading of the kernel interface list. It is useful to specify this with the -t option on the command line, because the first interface scan is done before reading the configuration file.

Table A–5 Protocol Significance Options

Option	Description
all	Turns on all of the options flags.
general	A shorthand notation for specifying both normal and route.
state	Traces state machine transitions in the protocols.

(continued on next page)

Table A–5 (Cont.) Protocol Significance Options

Option	Description
normal	Traces normal protocol occurrences. Abnormal protocol occurrences are always traced.
policy	Traces the application of protocol and user-specified policy to routes being imported and exported.
task	Traces system interface and processing associated with this protocol or peer.
timer	Traces timer usage by this protocol or peer.
route	Traces routing table changes for routes installed by this protocol or peer.

Note

Not all of these options apply to all of the protocols. In some cases, their use does not make sense (for instance, RIP does not have a state machine) and in some instances the requested tracing has not been implemented (such as RIP support of the policy option).

It is not possible to specify packet tracing from the command line because a global option for packet tracing would potentially create too much output.

When protocols inherit their tracing options from the global tracing options, tracing levels that do not make sense (such as parse, adv, and packet tracing options) are masked out.

Global tracing statements have an immediate effect, especially parsing options that affect the parsing of the configuration file. Tracing values inherited by protocols specified in the configuration file are initially inherited from the global options in effect as they are parsed, unless they are overridden by more specific options.

After the configuration file is read, tracing options that were not explicitly specified are inherited from the global options in effect at the end of the configuration file.

A.7.2 Packet Tracing

Every protocol has one or more options for tracing packets. All protocols allow the packets keyword to be used for tracing all packets sent and received by the protocol. Most protocols have other options for limiting tracing to a useful subset of packet types. These tracing options can be further controlled with the following modifiers:

detail	Specifies a more verbose format to provide more information about the contents of the packet. The detail option must be specified before send or recv. By default, packets are traced in a terse form of one or two lines.
send	Limits the tracing to packets sent received. If neither the send nor the recv option is specified, both sent and received packets are traced.

`recv` Limits the tracing to packets received. If neither the `send` nor the `recv` option is specified, both sent and received packets are traced.

Note

If a protocol allows several different types of packet tracing, modifiers can be applied to each individual type. Be aware, however, that within one tracing specification the trace flags are summed up, so specifying detail packets turns on full tracing for all packets.

A.8 Directive Statements

Directive statements provide direction to the GATED configuration language parser about included files and the directories in which these files reside. Directive statements are immediately acted upon by the parser. Other statements terminate with a semicolon (;), but directive statements terminate with a new line. The two directive statements are as follows:

- `%directory` *directory*

  Defines the directory in which the include files are stored. When it is used, GATED searches the directory identified by path name for any included files that do not have a fully qualified file name (do not begin with "/"). This statement does not change the current directory; it only specifies the prefix applied to included file names.

- `%include` *filename*

  Identifies an include file. The contents of the file is included in the TCPIP$GATED.CONF file at the point where the `%include` directive is located. If the file name is not fully qualified (does not begin with backslash (/), it is considered to be relative to the directory defined in the `%directory` directive. The `%include` directive statement causes the specified file to be parsed completely before resuming with this file. Nesting up to ten levels is supported. The maximum nesting level can be increased by changing the definition of FI_MAX in the `parse.h` file.

In a complex environment, segmenting a large configuration into smaller, more easily understood segments might be helpful, but one of the advantages of GATED is that it combines the configuration of several different routing protocols into a single file. Segmenting a small file unnecessarily complicates routing configurations.

A.9 Options Statements

The options statement allows specification of some global options. If used, options must appear before any other type of configuration statement in the TCPIP$GATED.CONF file.

The syntax for the options statement is as follows:

```
 options
[nosend]
[noresolv]
[gendefault [preference preference] [gateway gateway]]
[mark time]
;
```

The options list can contain one or more of the following options:

`gendefault` [preference *preference*] [gateway *gateway*]

When `gendefault` is enabled and a BGP or EGP neighbor is up, a default route with the special protocol default is created. This can be disabled per BGP/EGP group with the `nogendefault` option. By default, this route has a preference of 20. This route is normally not installed in the kernel forwarding table; it is only present so it can be announced to other protocols.

If a gateway is specified, the default route is installed in the kernel forwarding table with a next hop of the listed gateway.

Note that the use of the more general [generate default] option is preferred to the use of the `gendefault` option. The `gendefault` option may be removed in the future. See Section A.18.6 for more information.

`nosend`

Do not send any packets. This option makes it possible to run GATED on a live network to test protocol interactions without actually participating in the routing protocols. The packet traces in the GATED log can be examined to verify that GATED is functioning properly. This is useful for the RIP interface. This option does not apply to BGP and is not useful with EGP and OSPF.

`noresolv`

By default, GATED tries to resolve symbolic names into IP addresses by using the gethostbyname( ) and getnetbyname( ) library calls. These calls usually use the Domain Name System (DNS) instead of the host's local host and network tables. If there is insufficient routing information to send DNS queries, GATED deadlocks during startup. This option can be used to prevent these calls; symbolic names result in configuration file errors.

`mark time`

Specifying this option causes GATED to output a message to the trace log at the specified interval. This can be used to determine if GATED is still running.

A.10 Interface Statements

An interface is the connection between a router and one of its attached networks. A physical interface can be specified by interface name, by IP address, or by domain name (unless the network is an unnumbered point-to-point network). Multiple levels of reference in the configuration language allow identification of interfaces using a wildcard or interface type name. Be careful with the use of interface names because future versions of TCP/IP Services may allow more than one address per interface. The `interface_list` is a list of one or more interface names including wildcard names (names without a number) and names that may specify more than one interface or address, or the token `all` for all interfaces.

The syntax for the interfaces statement is as follows:

```
interfaces {
     options
          [strictinterfaces]
          [scaninterval time]
          [aliases-nexthop ( primary | lowestip | keepall )
          ;
     interface interface_list
          [preference preference]
          [down preference preference]
          [passive]
          [simplex]
          [reject]
          [blackhole]
          [ AS autonomoussystem ]
          ;
     define  address
          [broadcast address] | [pointtopoint  address]
          [netmask mask]
          [multicast]
          ;
        } ;
```

The options portion of the interfaces statement allows configuration of the following global options related to interfaces:

strictinterfaces — Indicates that it is a fatal error to refer to an interface in the configuration file that is not present when GATED is started and not listed in a define statement. Without strictinterfaces, a warning message is issued but GATED will continue.

scaninterval *time* — Specifies how often GATED scans the kernel interface list for changes. The default is every 15 seconds on most systems, and 60 seconds on systems that pass interface status changes through the routing socket (BSD 4.4). Note that GATED also scans the interface list on receipt of a SET GATED/CHECK_INTERFACES.

aliases-nexthop primary | lowestip | keepall — Specifies which address GATED will install as the next hop for interface routes. If you specify primary, the primary interface address (default) will be installed. If you specify lowestip, the address with the lowest IP address will be installed. If you specify keepall, all interface routes are kept in the kernel up to a maximum of RT_N_MULTIPATH routes. aliases-nexthop is a compile-time constant. aliases-nexthop is a global parameter that may be overridden for interfaces using the interface option.

The interface portion of the interfaces statement sets interface options on the specified interfaces. An interface list is all or a list of interface names (see Section A.10.1), domain names, or numeric addresses. Options available on this statement are:

preference *preference*	Sets the preference for routes to this interface when it is up and appears to be functioning properly. The default preference is 0.
down preference *preference*	Sets the preference for routes to this interface when GATED does not believe it to be functioning properly, but the kernel does not indicate it is down. The default value is 120.
passive	Prevents GATED from changing the preference of the route to this interface if it is not believed to be functioning properly due to lack of received routing information. The GATED daemon only performs this check if the interface is actively participating in a routing protocol.
simplex	Defines an interface as unable to hear its own broadcast packets. Some systems define an interface as simplex with the IFF_SIMPLEX flag; others require it to be specified in the configuration file. On simplex interfaces, a sender's own packets are assumed to have been looped back in software and are not used as an indication that the interface is functioning properly.
reject	Specifies that the address of the interface matching these criteria is used as the local address when installing reject routes in the kernel. Use reject only with systems based on BSD 4.3 Tahoe or earlier that have installed a reject/blackhole pseudo-interface.
blackhole	Specifies that the address of the interface matching these criteria is used as the local address when installing reject routes in the kernel. Use this only with systems based on BSD 4.3 Tahoe or earlier that have installed a reject/blackhole pseudo interface.

The define portion of the interfaces statement defines interfaces that might not be present when GATED is started so they may be referenced in the configuration file when strictinterfaces is defined. The following are valid define keywords:

broadcast *address*	Defines the interface as broadcast capable (for example, Ethernet or Token Ring) and specifies the broadcast address.
pointopoint *address*	Defines the interface as a point-to-point interface (for example, SLIP or PPP) and specifies the address on the local side. The first address on the define statement references the address of the host on the remote end of the interface, the address specified after this pointopoint keyword defines the address on the local side of the interface.
netmask *mask*	Specifies the subnet mask to be used on this interface. This is ignored on point-to-point interfaces.
multicast	Specifies that the interface is multicast capable.
AS autonomoussystem	Specifies the AS that will be used to create an AS path associated with the route created from the definition of this interface.

A.10.1 Interface Lists

An interface list is a list of references to interfaces or groups of interfaces. The following four methods, from most general to most specific, are available for referring to interfaces:

ALL	Refers to all available interfaces.
Interface name wildcard	Refers to all the interfaces of the same type. Interfaces consist of the device driver name and a unit number, for example, LE0. References to the name contain only alphabetic characters and match any interfaces that have the same alphabetic part.
Interface name	Refers to a specific interface, usually one physical interface. These are specified as an alphabetic part followed by a numeric part. This will match one specific interface. But be aware that on many systems, there can be more than one protocol (for example, IP) address on a given physical interface. For example, EF1 matches an interface named EF1, but not an interface named EF10.
Interface address	Matches one specific interface. The reference can be by protocol address (for example, 10.0.0.51) or by symbolic host name (for example, `nic.ddn.mil`). Note that a symbolic host name reference is only valid when it resolves to only one address. Use of symbolic host names is not recommended.

If many interface lists are present in the TCPIP$GATED.CONF file with more than one parameter, these parameters are collected at run time to create the specific parameter list for a given interface. If the same parameter is specified on more than one list, the parameters with the most specific interface are used.

For example, the following interface list is for a system with three interfaces, LE0, LE1, and DU0:

```
rip yes {
    interface all noripin noripout ;
    interface le ripin ;
    interface le1 ripout ;
} ;
```

In this example, RIP packets are accepted from interfaces LE0 and LE1, but nor from DU0. RIP packets are sent only on interface LE1.

A.10.1.1 Example of Current Define Statements for GATED

```
interfaces {
   define 192.168.12.5 broadcast 192.168.12.255 netmask 255.255.255.0 ;
   define 192.168.13.129 netmask 255.255.255.252 broadcast 192.168.13.131;

   # pointtopoint - is local side, 1st address is remote
   define  192.168.13.116 pointtopoint 192.168.13.114 multicast;
   };
```

- The first define statement has an Ethernet where you need to define the broadcast address as a /24.
- The second define statement shows how a /30 may be implemented in the define statement. The define tells GATED to treat the interface with a local address of 192.168.13.129, a netmask of 255.255.255.252, and a broadcast address of 192.168.13.131.
- The third define statement shows how a point-to-point interface is defined. The remote side of the point-to-point interface is specified first, and the local side (the one on this machine) is specified second.

A.10.2 IP Interface Addresses and Routes

The BSD 4.3 and later networking implementations allow the following four types of interfaces. Some implementations allow multiple protocol addresses per physical interface, but these are mostly based on BSD 4.3 RENO or later.

Loopback	This interface must have the address of 127.0.0.1. Packets sent to this interface are sent back to the originator. This interface is also used as an interface for implementing other features, such as reject and blackhole routes. Although a netmask is reported on this interface, it is ignored. It is useful to assign an additional address to this interface that is the same as the OSPF or BGP `routerid`; this allows routing to a system based on the router ID that will work if some interfaces are down.
Broadcast	This is a multiaccess interface capable of a physical level broadcast, such as Ethernet, Token Ring, and FDDI. This interface has an associated subnet mask and broadcast address. The interface route to a broadcast network is a route to the complete subnet.
Point-to-point	This is a tunnel to another host, usually on some sort of serial link. This interface has a local address and a remote address. The remote address must be unique among all the interface addresses on a given router. If a subnet mask is specified on a point-to-point interface, it is only used by RIP version 1 to determine which subnets may be propagated to the router on the other side of this interface.
Nonbroadcast multi-access (NBMA)	This type of interface is multiaccess, but not capable of broadcast, for example, frame relay and X.25. This type of interface has a local address and a subnet mask. (Not supported.)

The GATED daemon ensures that there is a route available to each IP interface that is configured and up. Normally this is done by the SET INTERFACE command that configures the interface; GATED also does it to ensure consistency.

For point-to-point interfaces, GATED installs some special routes. GATED installs a route to the local address pointing at the loopback interface with a preference of 110. This ensures that packets originating on this host destined for this local address are handled locally.

OSPF prefers to route packets for the local interface across the point-to-point link where they will be returned by the router on the remote end. This is used to verify operation of the link. Because OSPF installs routes with a preference of 10, these routes override the route installed with a preference of 110.

When the status of an interface changes, GATED notifies all the protocols, which take the appropriate action. The GATED daemon assumes that interfaces that are not marked UP do not exist.

The GATED daemon ignores any interfaces that have invalid data for the local, remote, or broadcast addresses or the subnet mask. Invalid data includes zeros in any field. The GATED daemon also ignores any point-to-point interface that has the same local and remote addresses; it assumes it is in some sort of loopback test mode.

A.11 Definition Statements

Definition statements are general configuration statements that relate to all of GATED, or at least to more than one protocol. The three definition statements are autonomoussystem, routerid, and martians. If used, autonomoussystem, routerid, and martians, must appear before any other type of configuration statement in TCPIP$GATED.CONF file.

A.11.1 Autonomous System Configuration

The statement autonomoussystem *as_number* [loops *number*]; sets the AS number of this router used by BGP EGP. The AS number is the official autonomous system number assigned to you by the Network Information Center (NIC).

The loops parameter is only for protocols supporting AS paths, such as BGP. It controls the number of times this autonomous system may appear in an AS path and defaults to 1 (one).

A.11.2 Router ID Configuration

The statement routerid *host*; sets the router identifier for use by the BGP and OSPF protocols. The default is the address of the first interface encountered by GATED. The address of a non-point-to-point interface is preferred over the local address of a point-to-point interface, and an address on a loopback interface that is not the loopback address (127.0.0.1) is most preferred.

A.11.3 Martian Configuration

Sometimes a misconfigured system sends out invalid destination addresses. These invalid addresses, called martians, are rejected by the routing software. A martian configuration defines a list of martian addresses from which all routing information is ignored. A martian configuration is structured as follows:

```
martians {
    host host [allow] ;
    network [allow] ;
    network mask mask [allow] ;
    network masklen number [allow] ;
    default [allow] ;
} ;
```

The martians *martian_list* statement adds martian addresses to a martian address list. Routing information will not be accepted from the addresses specified in this list.

You can specify the allow parameter to explicitly allow a subset of a range that was disallowed.

A.11.4 Sample Definition Statements

The following sample shows definition statements for a system:

```
options gendefault ;
autonomoussystem 249 ;
interface 128.66.12.2 passive ;
martians {
    0.0.0.26
};
```

The following list describes each statement in the example:

- The `options` statement tells the system to generate a default route when it peers with an EGP or BGP neighbor.
- The `autonomoussystem` statement tells GATED to use AS number 249 for EGP and BGP.
- The `interface` statement tells GATED not to mark interface 128.66.12.2 as down even if it sees no traffic.
- The `martians` statement prevents routes to 0.0.0.26 from ever being accepted.

A.12 Protocol Overview

Unicast routing protocols allow packets to be routed to one destination. All routing protocols determine the "best" route to each destination, and they distribute routing information among the systems on a network. Routing protocols are divided into two general groups: interior (or intradomain routing) protocols and exterior (or interdomain routing) protocols. GATED software combines management of the interior and exterior routing protocols in one software daemon.

A.12.1 Interior Routing Protocols

Interior protocols are used to exchange reachability information within an autonomous system (AS). They are referred to as a class by the acronym IGP. There are several interior protocols:

- RIP

  The Routing Information Protocol, Version 1 and Version 2, is the most commonly used interior protocol. RIP selects the route with the lowest metric as the best route. The metric is a hop count representing the number of gateways through which data must pass to reach its destination. The longest path that RIP accepts is 15 hops. If the metric is greater than 15, a destination is considered unreachable and GATED discards the route. RIP assumes the best route is the one that uses the fewest gateways i.e., the shortest path, not taking into account congestion or delay on route.

  The RIP version 1 protocol is described in RFC 1058 and the RIP version 2 protocol is described in RFC 1723.

- OSPF

  Open Shortest Path First is a link-state protocol. OSPF is better suited than RIP for complex networks with many routers. OSPF provides equal cost multipath routing.

  OSPF is described in RFC 1583, the MIB is defined in RFC 1253. Other related documents are RFC 1245, RFC 1246 and RFC 1370.

A.12.2 Exterior Routing Protocol

Exterior protocols are used to exchange routing information between autonomous systems. Exterior protocols are only required when an autonomous system must exchange routing information with another autonomous system. Routers within an autonomous system run an interior routing protocol like RIP. Only those gateways that connect an autonomous system to another autonomous system need to run an exterior routing protocol. There are two exterior protocols currently supported by GATED:

- EGP

  Exterior Gateway Protocol: Originally EGP reachability information was passed into ARPANET/MILNET "core" gateways where the best routes were chosen and passed back out to all connected autonomous systems. As the Internet moved toward a less hierarchical architecture, EGP, an exterior routing protocol which assumes a hierarchical structure, became less effective.

  The EGP protocol is described in RFC 827 and RFC 904.

- BGP

  Border Gateway Protocol is replacing EGP as the exterior protocol of choice. BGP exchanges reachability information between autonomous systems, but provides more capabilities than EGP. BGP uses path attributes to provide more information about each route as an aid in selecting the best route. Path attributes may include, for example, administrative preferences based on political, organizational, or security (policy) considerations in the routing decision. BGP supports nonhierarchical topologies and can be used to implement a network structure of equivalent autonomous systems.

  BGP version 1 is described in RFC 1105; version 2 in RFC 1163; version 3 in RFC 1267; and version 4 in RFC 1771. The version 3 MIB is described in RFC 1269. The three documents, RFC 1164, RFC 1268, and RFC 1772, describe the application of versions 2, 3, and 4 in the Internet. A protocol analysis of an experience with BGP version 3 is available in RFC 1265 and RFC 1266. RFC 1397 talks about advertising a default route in BGP version 2 and 3.

  BGP version 4 is described in RFC 1771. The BGP V4 MIB implemented by GATED is draft standard, but is scheduled to go to standard. Other references for BGP are: RFC 1997 (BGP Communities), RFC 1966 (BGP Route Reflection), RFC 1966 (BGP AS Confederations), and RFC 1403 (BGP–OSPF interaction). A useful application document is: RFC 1998 (An Application of the BGP Community Attribute in Multi-home Routing).

A.12.3 Router Discovery Protocol

The Router Discovery protocol is used to inform hosts of the availability of other hosts to which it can send packets. Router Discovery is used to supplement a statically configured default router. This is the preferred protocol for hosts to run. They are discouraged from wiretapping routing protocols. Router Discovery is described in RFC 1256

A.12.4 ICMP

On systems without the BSD routing socket, GATED listens to ICMP messages received by the system. Processing of ICMP redirect messages is handled by the redirect statement.

A.12.5 Redirect

The redirect code process ICMP or ISO redirects learned by monitoring ICMP messages, or via the routing socket on systems that support it. It processes the redirect request and decides whether to accept the redirect. If the redirect is accepted, a route is installed in the GATED routing table with the protocol redirect. Redirects are deleted from the routing table after 3 minutes.

A.12.6 Kernel Interface

Although the kernel interface is not technically a routing protocol, it has many characteristics of one, and GATED handles it similarly. The routes GATED chooses to install in the kernel forwarding table are those that will actually be used by the kernel to forward packets.

The add, delete and change operations that GATED must use to update the typical kernel forwarding table take a non-trivial amount of time. The time used does not present a problem for older routing protocols (RIP, EGP), which are not particularly time critical and do not easily handle very large numbers of routes anyway. The newer routing protocols (OSPF, BGP) have stricter timing requirements and are often used to process many more routes. The speed of the kernel interface becomes critical when these protocols are used.

A.12.7 Static Routes

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

A.13 The ICMP Statement

On systems without the BSD routing socket, GATED listens to ICMP messages received by the system. GATED currently supports router discovery as well as redirect. Processing of ICMP redirect messages is handled by the redirect statement.

Use the ICMP statement to trace the ICMP messages that GATED receives.

The following ICMP statement specifies the tracing options for ICMP.

```
icmp {
     traceoptions trace_options ;
     }

    traceoptions trace_options ;
```

A.13.1 Tracing Options

Packet tracing options (which may be modified with `detail` and `recv`):

`packets`	All ICMP packets received.
`redirect`	Only ICMP REDIRECT packets received.
`routerdiscovery`	Only ICMP ROUTER DISCOVERY packets received.
`info`	Only ICMP informational packets, which include mask request/response, info request/response, echo request/response and time stamp request/response.
`error`	Only ICMP error packets, which include time exceeded, parameter problem, unreachable and source quench.

A.14 Redirect Processing

The redirect code is passed ICMP or ISO redirects learned by monitoring ICMP messages, or via the routing socket on systems that support it. It processes the redirect request and decides whether to accept the redirect. If the redirect is accepted, a route is installed in the GATED routing table with the protocol redirect. Redirects are deleted from the routing table after 3 minutes.

If GATED determines that a redirect is not acceptable, it tries to figure out if the kernel forwarding table has been modified. On systems where ICMP messages are monitored this is accomplished by trying to second guess what the kernel would have done with the redirect. On systems with the routing socket, the kernel provides and indication of whether the redirect was accepted; GATED ignores redirects that were not processed.

If GATED has determined that the state of the kernel forwarding table has been changed, the necessary requests to the kernel are made to restore the correct state.

You cannot disable the processing of ICMP redirects, even when the system is functioning as a router. To ignore the effects of redirects, GATED must process each one and actively restore any changes it made to the kernel's state. Because of the mechanisms involved there will be windows where the effects of redirects are present in the kernel.

By default, GATED removes redirects when actively participating in an interior gateway protocol (RIP or OSPF). It is not possible to enable redirects once they have been automatically disabled. Listening to RIP in nobroadcast mode does not cause redirects to be ignored, nor does the use of EGP and BGP. Redirects must be manually configured off in these cases.

Note that in accordance with the latest IETF Router Requirements document, GATED insures that all ICMP net redirects are processed as host redirects. When an ICMP net redirect is accepted, GATED issues the requests to the kernel to make sure that the kernel forwarding table is updated to reflect a host redirect instead of a net redirect.

The redirect statement does not prevent the system from sending redirects, only from listening to them.

The redirect statement is formatted as follows:

```
redirect yes | no | on | off
[ {
    preference preference ;
    interface interface_list
        [ noredirects ] | [redirects ] ;
    trustedgateways gateway_list ;
    traceoptions trace_options ;
} ] ;
```

In the redirect statement:

- `preference` sets the preference for a route learned from a redirect. The default is 30.

- `interface` is the interface statement, which allows the enabling and disabling of redirects on an interface-by-interface basis. See Section A.10.1 for the description of the *interface_list*. The parameters are:
  - `noredirects`—Specifies that redirects received from the specified interface will be ignored. The default is to accept redirects on all interfaces.
  - `redirects`— This is the default. This argument may be necessary when noredirects are used on a wildcard interface descriptor.
- `trustedgateways` defines the list of gateways from which redirects will be accepted. The *gateway_list* is a list of host names or addresses. By default, all routers on the shared network(s) are trusted to supply redirects. But if the `trustedgateways` clause is specified, only redirects from the gateways in the list are accepted.

There are no redirect-specific tracing options. All nonerror messages are traced under the normal class.

A.15 The Router Discovery Protocol

The Router Discovery Protocol is an IETF standard protocol used to inform hosts of the existence of routers. It is intended to be used instead of having hosts wiretap routing protocols such as RIP. It is used in place of, or in addition to statically configured default routes in hosts.

The protocol is split into to portions, the server portion which runs on routers, and the client portion that runs on hosts. GATED treats these much like two separate protocols, only one of which may be enabled at a time.

A.15.1 The Router Discovery Server

The Router Discovery Server runs on routers and announces their existence to hosts. It does this by periodically multicasting or broadcasting a **Router Advertisement** to each interface on which it is enabled. These Router Advertisements contain a list of all the routers addresses on a given interface and their preference for use as a default router.

Initially these Router Advertisements occur every few seconds, then fall back to every few minutes. In addition, a host may send a **Router Solicitation** to which the router will respond with a unicast Router Advertisement (unless a multicast or broadcast advertisement is due momentarily).

Each Router Advertisement contains a Advertisement Lifetime field, which indicates for how long the advertised addresses are valid. This lifetime is configured such that another Router Advertisement will be sent before the lifetime has expired. A lifetime of zero is used to indicate that one or more addresses are no longer valid.

On systems supporting IP multicasting, the Router Advertisements are by default sent to the all-hosts multicast address 224.0.0.1. However, the use of broadcast may be specified. When Router Advertisements are being sent to the all-hosts multicast address, or an interface is configured for the limited-broadcast address 255.255.255.255, all IP addresses configured on the physical interface are included in the Router Advertisement. When the Router advertisements are being sent to a net or subnet broadcast, only the address associated with that net or subnet is included.

The Router Discovery Server syntax is as follows:

```
routerdiscovery server yes | no | on | off [ {
    traceoptions trace_options ;
    interface interface_list
        [ minadvinterval time ]
        [ maxadvinterval time ]
        [ lifetime time ]
        ;
    address interface_list
        [ advertise ] | [ ignore ]
        [ broadcast ] | [ multicast ]
        [ ineligible ] | [ preference preference ]
        ;
} ] ;
```

The Router Discovery Server syntax includes the following:

- traceoptions specifies the Router Discovery tracing options (see Section A.15.3).
- interface specifies the parameters that apply to physical interfaces. Note a slight difference in convention from the rest of GATED, interface specifies just physical interfaces (such as LE0, EF0 and EN1), while address specifies protocol (in this case IP) addresses.

  The interface parameters are:

  - maxadvinterval specifies the maximum time allowed between sending broadcast or multicast Router Advertisements from the interface. Must be no less than 4 and no more than 30:00 (30 minutes or 1800 seconds). The default is 10:00 (10 minutes or 600 seconds).
  - minadvinterval specifies the minimum time allowed between sending unsolicited broadcast or multicast Router Advertisements from the interface. Must be no less than 3 seconds and no greater than maxadvinterval. The default is 0.75 * maxadvinterval.
  - lifetime specifies the life time of addresses in a Router Advertisement. Must be no less than maxadvinterval and no greater than 2:30:00 (two hours, thirty minutes or 9000 seconds). The default is 3 * maxadvinterval.
- address specifies the parameters that apply to the specified set of addresses on this physical interfaces. Note a slight difference in convention from the rest of GATED, interface specifies just physical interfaces (such as LE0, EF0 and EN1), while address specifies protocol (in this case IP) addresses.

  The address parameters are:

  - advertise, which Specifies that the specified addresses should be included in Router Advertisements. This is the default.
  - ignore, which specifies that the specified addresses should not be included in Router Advertisements.
  - broadcast, which specifies that the given addresses should be included in a broadcast Router Advertisement because this system does not support IP multicasting, or some hosts on attached network do not support IP multicasting. It is possible to mix addresses on a physical interface such that some are included in a broadcast Router Advertisement and some are included in a multicast Router Advertisement. broadcast is the default if the router does not support IP multicasting.

- multicast, which specifies that the given addresses should only be included in a multicast Router Advertisement. If the system does not support IP multicasting the addresses will not be included. If the system supports IP multicasting, the default is to include the addresses in a multicast Router Advertisement if the given interface supports IP multicasting, if not the addresses will be included in a broadcast Router Advertisement.
- preference, which specifies the preferability of the addresses as a default router address, relative to other router addresses on the same subnet. A 32-bit, signed, twos-complement integer, with higher values meaning more preferable. Note that hex 80000000 may only be specified as ineligible. The default is 0.
- ineligible, which specifies that the given addresses will be assigned a preference of (hex 80000000) which means that it is not eligible to be the default route for any hosts.

  This is useful when the addresses should not be used as a default route, but are given as the next hop in an ICMP redirect. This allows the hosts to verify that the given addresses are up and available.

A.15.2 The Router Discovery Client

A host listens for Router Advertisements via the all-hosts multicast address (224.0.0.2), If IP multicasting is available and enabled, or on the interface's broadcast address. When starting up, or when reconfigured, a host may send a few Router Solicitations to the all-routers multicast address, 224.0.0.2, or the interface's broadcast address.

When a Router Advertisement with non-zero lifetime is received, the host installs a default route to each of the advertised addresses. If the preference ineligible, or the address is not on an attached interface, the route is marked unusable but retained. If the preference is usable, the metric is set as a function of the preference such that the route with the best preference is used. If more than one address with the same preference is received, the one with the lowest IP address will be used. These default routes are not exportable to other protocols.

When a Router Advertisement with a zero lifetime is received, the host deletes all routes with next-hop addresses learned from that router. In addition, any routers learned from ICMP redirects pointing to these addresses will be deleted. The same will happen when a Router Advertisement is not received to refresh these routes before the lifetime expires.

The Router Discovery Client syntax is as follows:

```
routerdiscovery client yes | no | on | off [ {
    traceoptions trace_options ;
    preference preference ;
    interface interface_list
        [ enable ] | [ disable ]
        [ broadcast ] | [ multicast ]
        [ quiet ] | [ solicit ]
        ;
} ] ;
```

In the Router Discovery Client statement:

- traceoptions specifies the tracing options for OSPF (see Section A.15.3).

- preference specifies the preference of all Router Discovery default routes. The default is 55.
- interface specifies the parameters that apply to physical interfaces. Note a slight difference in convention from the rest of GATED, interface specifies just physical interfaces (such as LE0, EF0 and EN1). The Router Discovery Client has no parameters that apply only to interface addresses.

  The interface parameters that apply to physical interfaces are:

  - enable, which specifies that Router Discovery should be performed on the specified interfaces. This is the default.
  - disable, which specifies that Router Discovery should not be performed on the specified interfaces.
  - broadcast, which specifies that Router Solicitations should be broadcast on the specified interfaces. This is the default if IP multicast support is not available on this host or interface.
  - multicast, which specifies that Router Solicitations should be multicast on the specified interfaces. If IP multicast is not available on this host and interface, no solicitation will be performed. The default is to multicast Router Solicitations if the host and interface support it, otherwise Router Solicitations are broadcast.
  - quiet, which specifies that no Router Solicitations will be sent on this interface, even though Router Discovery will be performed.
  - solicit, which specifies that initial Router Solicitations will be sent on this interface. This is the default.

A.15.3 Tracing Options

The Router Discovery Client and Server support the state trace flag, which traces various protocol occurrences.

The Router Discovery Client and Server do not directly support any packet tracing options, tracing of router discovery packets is enabled with the ICMP statement.

A.16 The Kernel Statement

While the kernel interface is not technically a routing protocol, it has many of the characteristics of one, and GATED handles it similarly to one. The routes GATED chooses to install in the kernel forwarding table are those that will actually be used by the kernel to forward packets.

The add, delete and change operations GATED must use to update the typical kernel forwarding table take a non-trivial amount of time. This does not present a problem for older routing protocols (RIP, EGP), which are not particularly time critical and do not easily handle very large numbers of routes anyway. The newer routing protocols (OSPF, BGP) have stricter timing requirements and are often used to process many more routes. The speed of the kernel interface becomes critical when these protocols are used.

To prevent GATED from locking up for significant periods of time installing large numbers of routes (up to a minute or more has been observed on real networks), the processing of these routes is now done in batches. The size of these batches may be controlled by the tuning parameters described below, but normally the default parameters will provide the proper functionality.

During normal shutdown processing, GATED normally deletes all the routes it has installed in the kernel forwarding table, except for those marked with **retain**. Optionally, GATED can leave all routes in the kernel forwarding table by not deleting any routes. In this case changes will be made to insure that routes with a **retain** indication are installed in the table. This is useful on systems with large numbers of routes as it prevents the need to re-install the routes when GATED restarts. This can greatly reduce the time it takes to recover from a restart.

A.16.1 Forwarding Tables and Routing Tables

The table in the kernel that controls the forwarding of packets is a **forwarding table**, also known as a **forwarding information base**, or FIB. The table that GATED uses internally to store routing information it learns from routing protocols is a **routing table**, also known as a **routing information base**, or RIB. The routing table is used to collect and store routes from various protocols. For each unique combination of network and mask an active route is chosen, this route will be the one with the best (numerically smallest) preference. All the active routes are installed in the kernel forwarding table. The entries in this table are what the kernel actually uses to forward packets.

A.16.2 Updating the Forwarding Table

There are two main methods of updating the kernel FIB, the `ioctl( )` interface and the routing socket interface. Their various characteristics are described here.

A.16.2.1 Updating the Forwarding Table with the ioctl Interface

The `ioctl` interface to the forwarding table was introduced in BSD 4.3. This is a one-way interface; it only allows GATED to update the kernel forwarding table. It has several other limitations:

- Fixed subnet masks

  The BSD 4.3 networking code assumed that all subnets of a given network had the same subnet mask. This limitation is enforced by the kernel. The network mask is not stored in the kernel forwarding table, but determined when a packet is forwarded by searching for interfaces on the same network.

- One way interface

  GATED is able to update the kernel forwarding table, but it is not aware of other modifications of the forwarding table. GATED is able to listen to ICMP messages and guess how the kernel has updated the forwarding table with response to ICMP redirects.

- Blind updates

  GATED is not able to detect changes to the forwarding table resulting from the use of the ROUTE command. Use of the ROUTE command on systems that use the `ioctl( )` interface is strongly discouraged while GATED is running.

- Changes not supported

  In all known implementations, there is no change operation supported, to change a route that exists in the kernel, the route must be deleted and a new one added.

A.16.2.2 Updating the Forwarding Table with the Routing Socket Interface

The routing socket interface to the kernel forwarding table was introduced in BSD 4.3 Reno, widely distributed in BSD 4.3 Net/2 and improved in BSD 4.4. This interface is simply a socket, similar to a UDP socket, on which the kernel and GATED exchange messages. It has several advatages over the `ioctl( )` interface:

- Variable subnet masks

  The network mask is passed to the kernel explicitly. This allows different masks to be used on subnets of the same network. It also allows routes with masks that are more general than the natural mask to be used. This is known as classless routing.

- Two way interface

  Not only is GATED able to change the kernel forwarding table with this interface, but the kernel can also report changes to the forwarding table to GATED. The most interesting of these is an indication that a redirect has modified the kernel forwarding table; this means that GATED no longer needs to monitor ICMP messages to learn about redirects. Plus, there is an indication of whether the kernel processed the redirect, GATED can safely ignore redirect messages that the kernel did not process.

- Updates visible

  Changes to the routing table by other processes, including the route command are received via the routing socket. This allows GATED to insure that the kernel forwarding table is synchronized with the routing table. Also, it allows the system administrator to perform some operations with the ROUTE command while GATED is running.

- Changes supported

  There is a functioning change message that allows routes in the kernel to be atomically changed. Some early verions of the routing socket code had bugs in the change message processing. There are compilation time and configuration time options that cause delete and add sequences to be used instead of change messages.

- Expandable

  New levels of kernel and GATED communications may be added by adding new message types.

A.16.3 Reading the Forwarding Table

When GATED starts up it reads the kernel forwarding table and installs corresponding routes in the routing table. These routes are called remnants and are timed out after a configured interval (which defaults to 3 minutes), or as soon as a more attractive route is learned. This allows forwarding to occur during the time it takes the routing protocols to start learning routes.

There are three main methods for reading the forwarding table from the kernel:

- Reading forwarding table with KMEM

  On many systems, especially those based on BSD 4.3, GATED must have knowledge of the kernel's data structures to read the current state of forwarding table. This method is slow and subject to error if the kernel forwarding table is updated while GATED is reading it. This can happen if the system administrator uses the ROUTE command, or an ICMP redirect message is received while GATED is starting up.

Due to an oversight, some systems (such as OSF/1) that are based on BSD 4.3 Reno or later, do not have the getkerninfo( ) system call described below, which allows GATED to read routes from the kernel without knowing about kernel internal structures. On these systems it is necessary to read the kernel radix tree from kernel memory. This is even more error-prone than reading the hash based forwding table.

- Reading the forwarding table via `getkerninfo` or `sysctl`

  Besides the routing socket, BSD 4.3 Reno introduced the `getkerninfo( )` system call. This call allows a user process (of which GATED is one) to read information from the kernel without knowledge of the kernel data structures. In the case of the forwarding table, it is returned to GATED atomically as a series of routing socket messages. This prevents the problem associated with the forwarding table changing while GATED is in the process of reading it.

  BSD 4.4 changed the `getkerninfo( )` interface into the `sysctl( )` interface, which takes different parameters, but otherwise functions identically.

- Reading the forwarding table via OS specific methods

  Some operating systems define their own method of reading the kernel forwarding table.

A.16.4 Reading the Interface List

The kernel support subsystem of GATED is resposible for reading the status of the kernel's physical and protocl interfaces periodically. GATED detects changes in the interface list and notifies the protocols so they can start or stop instances or peers. The interface list is read one of two ways:

- Reading the interface list with SIOCGIFCONF

  On systems based on BSD 4.3, 4.3 Reno and 4.3 Net/2 the SIOCGIFCONF `ioctl` interface is used to read the kernel interface list. Using this method, a list of interfaces and some basic information about them is return by the SIOCGIFCONF call. Other information must be learned by issuing other `ioctl`s to learn the interface network mask, flags, MTU, metric, destination address (for point-to-point interfaces) and broadcast address (for broadcast capable interfaces).

  GATED reads rereads this list every 15 second looking for changes. When the routing socket is in use, it also rereads it whenever a messages is received indicating a change in routing configuration. Receipt of a SIGUSR2 signal also causes GATED to reread the list. This interval may be explicitly configured in the interface configuration.

- Reading the interface list with `sysctl`

  BSD 4.4 added the ability to read the kernel interface list via the `sysctl` system call. The interface status is returned atomically as a list of routing socket messages that GATED parses for the required information.

  BSD 4.4 also added routing socket messsages to report interface status changes immediately. This allows GATED to react quickly to changes in interface configuration.

  When this method is in use, GATED rereads the interface list only once a minute. It also rereads it on routing table changes indications and when a SIGUSR2 is received. This interval may be explicitly configured in the interface configuration.

A.16.5 Reading Interface Physical Addresses

Later version of the getkerninfo( ) and sysctl( ) interfaces return the interface physical addresses as part of the interface information. On most systems where this information is not returned, GATED scans the kernel physical interface list for this information for interfaces with IFFBROADCAST set, assuming that their drivers are handled the same as Ethernet drivers. On some systems, system specific interfaces are used to learn this information.

The interface physical addresses are useful for IS-IS. For IP protocols, they are not currently used, but they may be used in the future.

A.16.6 Reading Kernel Variables

At startup, GATED reads some special variables out of the kernel. This is usually done with the `nlist` (or `kvm_nlist`) system call, but some systems use different methods.

The variables read include the status of UDP checksum creation and generation, IP forwarding and kernel version (for informational purposes). On systems where the routing table is read directly from kernel memory, the root of the hash table or radix tree routing table is read. On systems where interface physical addresses are not supplied by other means, the root of the interface list is read.

A.16.7 Special Route Flags

The later BSD based kernel support the special route flags described in the following list:

- RTF_REJECT

  Instead of forwarding a packet like a normal route, routes with RTF_REJECT cause packets to be dropped and unreachable messages to be sent to the packet originators. This flag is only valid on routes pointing at the loopback interface.

- RTF_BLACKHOLE

  Like the RTF_REJECT flag, routes with RTF_BLACKHOLE cause packets to be dropped, but unreachable messages are not sent. This flag is only valid on routes pointing at the loopback interface.

- RTF_STATIC

  When GATED starts, it reads all the routes currently in the kernel forwarding table. Besides interface routes, it usually marks everything else as a remnant from a previous run of GATED and deletes it after a few minutes. This means that routes added with the ROUTE command will not be retained after GATED has started.

  To fix this the RTF_STATIC flag was added. When the route command is used to install a route that is not an interface route it sets the RTF_STATIC flag. This signals to GATED that the specified route was added by the systems administrator and should be retained.

A.16.8 Kernel Configuration Syntax

The kernel configuration syntax is as follows:

```
kernel {
    options
        [ nochange ]
        [ noflushatexit ]
        ;
    routes number ;
    flash
        [ limit number ]
        [ type interface | interior | all ]
        ;
    background
        [ limit number ]
        [ priority flash | higher | lower ]
        ;
    traceoptions trace_options ;
} ;
```

In the kernel configuration syntax:

- `options` specifies kernel options. Valid options are:
  - `nochange`, which, on systems supporting the routing socket, ensures that changes operations will not be performed, only deletes and adds. This is useful on early versions of the routing socket code where the change operation was broken.
  - `noflushatexit`, which specifies that during normal shutdown processing GATED deletes all routes from the kernel forwarding table that do not have a retain indication. The `noflushatexit` option prevents route deletions at shutdown. Instead, routes are changed and added to make sure that all the routes marked with retain get installed.

    This is useful on systems with thousands of routes. Upon startup GATED will notice which routes are in the kernel forwarding table and not have add them back.
- `routes` specifies the routes number. On some systems kernel memory is at a premium. With this parameter a limit can be placed on the maximum number of routes GATED will install in the kernel. Normally GATED adds/changes/deletes routes in interface/internal/external order, for example, it queues interface routes first, followed by internal routes, followed by external routes, and processes the queue from the beginning. If a this parameter is specified and the limit is hit, GATED does two scans of the list instead. On the first scan it does deletes, and also deletes all changed routes, turning the queued changes into adds. It then rescans the list doing adds in interface/internal/external order until it hits the limit again. This will tend to favor internal routes over external routes. The default is not to limit the number of routes in the kernel forwarding table.
- `flash` specifies that a route has changed. The process of notifying the protocols is called a flash update. The kernel forwarding table interface is the first to be notified. Normally a maximum of 20 interface routes may be processed during one flash update. The `flash` command allows tuning of the following parameters:
  - `limit` *`number`*, which specifies the maximum number of routes which may be processed during one flash update. The default is 20. A value of -1

will cause all pending route changes of the specified type to be processed during the flash update.

  - type, which specifies the type of routes that will be processed during a flash update. Interior specifies that interior routes will also be installed (see Section A.12.1). all specifies the inclusion of exterior routes as well (see Section A.12.2). The default is interface, which specifies that only interface routes will be installed during a flash update.

    Specifying flash limit -1 all causes all routes to be installed during the flash update; this mimics the behavior of previous versions of GATED.

- background specifies that the remaining routes are processed in batches in the background, that is, when no routing protocol traffic is being received. Normally, 120 routes are installed at a time to allow other tasks to be performed and the background processing is done at lower priority than flash updates the following parameters allow tuning of these parameters:

  - limit, which specifies the number of routes which may be processed at during one batch. The default is 120.
  - priority, which specifies the priority of the processing of batches of kernel updates in relationship to the flash update processing. The default is lower, which means that flash updates are processed first. To process kernel updates at the same priority as flash updates, specify flash. To process kernel updates at a higher priority, use higher.

A.16.9 Kernel Tracing Options

While the kernel interface is not technically a routing protocol, in many cases it is handled as one. You can enter the following two symbols from the command line because the code that uses them is executed before the trace file is parsed.

symbols	Symbols read from the kernel, by nlist( ) or similar interface.
iflist	Interface list scan. This option is useful when entered from the command line as the first interface list scan is performed before the configuration file is parsed.

The following tracing options can be specified only in the configuration file. They are not valid from the command line.

remnants	Routes read from the kernel when GATED starts.
request	Requests by GATED to Add/Delete/Change routes in the kernel forwarding table.

Use the following general option and packet-tracing options to systems that use the routing socket to exchange routing information with the kernel. They do not apply to systems that use the old BSD 4.3 ioctl( ) interface to the kernel.

- info

  Records informational messages received from the routing socket, such as TCP lossage, routing lookup failure, and route resolution requests. GATED does not currently do processing on these messages, just logs the information if requested.

Packet tracing options (which may be modified with detail, send, and recv) specify the types of message and include:

- routes

  Routes exchanged with the kernel, including Add/Delete/Change messages and Add/Delete/Change messages received from other processes.

- redirect

  Redirect messages received from the kernel.

- interface

  Interface status messages received from the kernel. These are only supported on systems with networking code derived from BSD 4.4.

- other

  Other messages received from the kernel, including those mentioned in the info type above.

A.17 Static Routes Statements

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

There are two forms of static statements. One defines a static route through a gateway. The other is used to support multiple network addresses on a single interface.

To define a static route through a gateway, use the following syntax:

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

} ;
```

host *host* | default | *network* [(mask *mask*) | (masklen *number*)] gateway *gateway_list*)

This is the most general form of the static statement. It defines a static route through one or more gateways. Static routes are installed when one or more of the gateways listed are available on directly attached interfaces. If more than one eligible gateway is available, these are limited by the number of multipath destinations supported (this compile-time parameter is currently almost always one on UNIX).

To define a static for multiple network addresses on an interface, use the following syntax:

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

network [(mask *mask*) | (masklen *number*)] interface *interface*

This sytax is used to define a static interface route which is used for primitive support of multiple network addresses on one interface.

The parameters for the static route statement are as follows:

- interface *interface_list*

  When interface is specified, gateways are only considered valid when they are on one of these interfaces. See Section A.10.1 for the description of the *interface_list*.

- preference *preference*

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

- retain

  Normally, GATED removes all routes except interface routes from the kernel forwarding table during a graceful shutdown. The retain option may be used to prevent specific static routes from being removed. retain insures that some routing is available when GATED is not running.

- reject

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

- blackhole

  A blackhole route is the same as a reject route except that unreachable messages are not supported. Specifying blackhole causes this route to be installed as a blackhole route.

- noinstall

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

A.18 Control Statements

The control statements are used to define:

- Route filtering, described in Section A.18.1
- Matching AS paths, as described in Section A.18.2
- Importing routes, as described in Section A.18.3
- Exporting routes, as described in Section A.18.4
- The source of exported routes, as described in Section A.18.5
- Route aggregation, as described in Section A.18.6

A.18.1 Route Filtering

Routes are filtered by specifying configuration language that will match a certain set of routes by destination, or by destination and mask. Among other places, route filters are used on martians, and in import and export statements.

The action taken when no match is found is dependent on the context, for instance import and export route filters assume an all reject ; at the end a list.

A route will match the most specific filter that applies. Specifying more than one filter with the same destination, mask and modifiers will generate an error.

Filtering syntax:

```
network [ exact | refines | between number and number ]
network mask mask [ exact | refines | between number and number ]
network masklen number [ exact | refines | between number and number ]
all
default
host host
```

These are all the possible formats for a route filter. Not all of these formats are available in all places, for instance the host and default formats are not valid for martians.

In most cases it is possible to specify additional parameters relevent to the context of the filter. For example, on a martian statement it is possible to specify the allow keyword, on an import statement you can specify a preference, and on an export you can specify a metric.

Each control statement is described in the following list:

- network [exact | refines | between *lownumber* and *highnumber*] network mask mask [exact | refines | between *lownumber* and *highnumber*]] network masklen number [exact | refines | between *lownumber* and *highnumber*]

  Matching usually requires both an address and a mask, although the mask is implied in the shorthand forms listed below. These three forms vary in how the mask is specified. In the first form, the mask is implied to be the natural mask of the network. In the second, the mask is explicitly specified. In the third, the mask is specified by the number of contiguous one bits.

  If no additional parameters are specified, any destination that falls in the range given by the network and mask is matched, the mask of the destination is ignored. If a natural network is specified, the network, any subnets, and any hosts will be match. The two optional modifiers cause the mask of the destination to be considered also:

exact	Specifies that the mask of the destination must match the supplied mask exactly. This is used to match a network, but no subnets or hosts of that network.
refines	Specifies that the mask of the destination must be more specified (for example, longer) than the filter mask. This is used to match subnets or hosts of a network, but not the network.
between *lownumber* and *highnumber*	Specifies that the mask of the destination must be as or more specific (for example, as long as or longer) than the lower limit (*lownumber*) and no more specific (for example, as long as or shorter) than the upper limit (*highnumber*). Note that exact and refines are both special cases of between.

- all

  This entry matches anything. It is equivalent to:

  ```
  0.0.0.0 mask 0.0.0.0
  ```

- default

  Matches the default route. To match, the address must be the default address and the mask must be all zeros. This is equivalent to:

  ```
  0.0.0.0 mask 0.0.0.0 exact
  ```

- host *host*

  Matches the specific host. To match, the address must exactly match the specified host and the network mask must be a host mask (i.e. all ones). This is equivalent to:

  ```
  host mask 255.255.255 exact
  ```

A.18.2 Matching AS Paths

An AS path includes a list of autonomous systems that routing information has passed through to get to a specified router, and an indicator of the origin of this list. This routing information can be used to prefer one path to a destination network over another. The primary method for preferring a route with GATED is to specify a list of filters to be applied to AS paths when importing and exporting routes.

Each autonomous system that a route passed through prepends its AS number to the beginning of the AS path.

AS path regular expressions are defined in RFC 1164.

A.18.2.1 AS Path-Matching Syntax

An AS path is matched using the following syntax.

```
aspath aspath_regexp origin ( [ any ] ) | [ igp ] | [egp ] | [ incomplete ] )
aspath aspath_regexp
```

aspath specifies that an AS matching the *aspath_regexp* with the specified origin is matched.

```
origin ( [ any ] | [ igp ] | [ egp ] | [ incomplete ] )
```

An `origin` of `igp` indicates the route was learned from an intradomain routing protocol and is most likely complete. An origin of `egp` indicates the route was learned from an interdomain routing protocol that does not support AS paths (EGP, for example), and the path is most likely not complete. When the path information is definitely not complete, an origin of `incomplete` is used. An origin of `any` can be used for any origin.

A.18.2.2 AS Path Regular Expressions

Technically, an AS path regular expression is a regular expression with the alphabet being the set of AS numbers. An AS path regular expression is composed of one or more AS paths expressions. An AS path expressions is composed of AS path terms and AS path operators.

A.18.2.3 AS Path Terms

An AS path term is one of the following three objects:

- *autonomous_system*

  Specifies any valid autonomous system number, from one through 65534 inclusive.

- dot (.)

  Matches any autonomous system number.

- `( aspath_regexp )`

  Group subexpressions in parenthese. An operator, such as * or ? works on a single element or on a regular expression enclosed in parentheses.

A.18.2.4 AS Path Operators

An AS path operator is one of the following:

- *aspath_term* {*m*,*n*}

  Indicates a regular expression followed by {*m*,*n*} (where *m* and *n* are nonnegative integers and *m* <= *n*) specifies at least *m* and at most *n* repetitions.

- *aspath_term* {*m*}

  Indicates a regular expression followed by {*m*}. When *m* is a positive integer, the expression specifies exactly *m* repetitions.

- *aspath_term* {*m*,}

  Indicates a regular expression followed by {*m*,} (where *m* is a positive integer), and specifies *m* or more repetitions.

- *aspath_term* *

  Indicates an AS path term followed by asterisk (*), specifying zero or more repetitions. This is shorthand for {0,}.

- *aspath_term* +

  Indicates a regular expression followed by plus sign (+), specifying one or more repetitions. This is shorthand for {1,}.

- *aspath_term* ?

  Indicates a regular expression followed by question mark (?), specifying zero or one repetition. This is shorthand for {0,1}.

- *aspath_term* | *aspath_term*

  Matches the AS term on the left, or the AS term on the right.

A.18.3 The Import Statement

Importation of routes from routing protocols and installation of the routes in GATED'S routing database is controlled by import statements. The format of an import statement varies depending on the source protocol.

A.18.3.1 Specifying Preferences

You can specify one of the following keywords to control how routes compete with other protocols:

`restrict`

`preference` *preference*

In these statements:

- `restrict` specifies that the routes are not desired in the routing table. In some cases this means that the routes are not installed in the routing table. In others it means that they are installed with a negative preference; this prevents them from becoming active so they will not be installed in the forwarding table, or exported to other protocols.
- `preference` specifies the preference value used when comparing this route to other routes from other protocols. The route with the lowest preference available at any given route becomes the active route, is installed in the forwarding table, and is eligible to be exported to other protocols. The default preferences are configured by the individual protocols.

A.18.3.2 Route Filters

All the formats allow route filters described in this section. When no route filtering is specified (that is, when `restrict` is specified on the first line of a statement), all routes from the specified source will match that statement. If any filters are specified, only routes that match the specified filters will be imported. That is, if any filters are specified, a statement like `all restrict ;` is assumed at the end of the list.

```
network [ exact | refines | between number and number]
network mask mask [exact | refines | between number and number ]
network masklen number [ exact | refines | between number and number ]
default
host host
```

A.18.3.3 Importing Routes from BGP and EGP

Use the following syntax to define importing routes from BGP and EGP:

```
import proto bgp | egp autonomoussystem autonomous_system
    [ aspath-opt ] restrict ;
import proto bgp | egp autonomoussystem autonomous_system
     [ aspath-opt ] [ preference preference ] {
    route_filter [ restrict | ( preference preference ) ] ;
} ;
```

```
import proto bgp aspath aspath_regexp
    origin any | ( [ igp ] [egp ] [ incomplete ] )
     [ aspath-opt ] restrict ;
import proto bgp aspath aspath_regexp
    origin any | ( [ igp ] [egp ] [ incomplete ] )
     [ aspath-opt ] [ preference preference ] {
    route_filter [ restrict | ( preference preference ) ] ;
} ;
```

EGP importation may be controlled by autonomous system. BGP also supports controlling propagation by the use of an AS path regular expressions, which are documented in the section on Matching AS paths. Note that EGP and BGP versions 2 and 3 only support the propagation of natural networks, so the host and default route filters are meaningless. BGP version 4 supports the propagation of any destination along with a contiguous network mask.

The aspath-opt option allows the specification of import policy based on the path attributes found in the BGP update. (The option is not usable with EGP.) If multiple communities are specified in the aspath-opt option, only updates carrying all of the specified communities will be matched. If none is specified, only updates lacking the community attribute will be matched.

Note that it is quite possible for several BGP import clauses to match a given update. If more than one clause matches, the first matching clause will be used; all later matching clauses will be ignored. For this reason, it is generally desirable to order import clauses from most to least specific. An import clause without an aspath-opt option will match any update with any communities or none.

EGP and BGP both store any routes that were rejected implicitly by not being metioned in a route filter, or explicitly with the restrict keyword in the routing table with a negative preference. A negative preference prevents a route from becoming active, which prevents it from being installed in the forwarding table, or exported to other protocols. This aleviates the need to break and re-establish a session upon reconfiguration if importation policy is changed.

A.18.3.4 Importing Routes from RIP and Redirects

Use the following syntax to define importing routes from RIP and redirect routes:

```
import proto rip | hello | redirect
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    restrict ;
import proto rip | hello | redirect
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    [ preference preference ] {
    route_filter [ restrict | ( preference preference ) ] ;
} ;
```

The importation of RIP and redirect routes may be controlled by any of protocol, source interface and source gateway. If more than one is specified, they are processed from most general (protocol) to most specific (gateway).

RIP does not support the use of preference to choose between routes of the same protocol. That is left to the protocol metrics. These protocols do not save routes that were rejected since they have short update intervals.

A.18.3.5 Importing Routes from OSPF

Use the following syntax to define importing routes from OSPF:

```
import proto ospfase [ tag ospf_tag ] restrict ;
import proto ospfase [ tag ospf_tag ]
    [ preference  preference ] {
     route_filter [ restrict | ( preference  preference ) ] ;
} ;
```

Due to the nature of OSPF, only the importation of ASE routes may be controlled. OSPF intra- and inter-area routes are always imported into the GATED routing table with a preference of 10. If a tag is specified, the import clause will only apply to routes with the specified tag.

It is only possible to restrict the importation of OSPF ASE routes when functioning as an AS border router. This is accomplished by specifying an export ospfase clause. Specification of an empty export clause may be used to restrict importation of ASEs when no ASEs are being exported.

Like the other interior protocols, preference can not be used to choose between OSPF ASE routes, that is done by the OSPF costs. Routes that are rejected by policy are stored in the table with a negative preference.

A.18.4 The Export Statement

The import statement controls which routes received from other systems are used by GATED; the export statement controls which routes are advertised by GATED to other systems. Like the import statement, the syntax of the export statement varies slightly per protocol. The syntax of the export statement is similar to the syntax of the import statement, and the meanings of many of the parameters are identical. The main difference between the two is that while route importation is just controlled by source information, route exportation is controlled by both destination and source.

The outer portion of a given export statement specifies the destination of the routing information you are controlling. The middle portion restricts the sources of importation that you wish to consider. And the innermost portion is a route filter used to select individual routes.

A.18.4.1 Specifying Metrics

The most specific specification of a metric is the one applied to the route being exported. The values that may be specified for a metric depend on the destination protocol that is referenced by this export statement.

```
restrict
metric metric
```

In this syntax:

- `restrict` specifies that nothing should be exported. If specified on the destination portion of the export statement it specifies that nothing at all should be exported to this destination. If specified on the source portion it specifies that nothing from this source should be exported to this destination. If specified as part of a route filter it specifies that the routes matching that filter should not be exported.
- `metric` *`metric`* specifies the metric to be used when exporting to the specified destination.

A.18.4.2 Route Filters

All the formats allow route filters as shown in the following example. See the section on route filters for a detailed explaination of how they work. When no route filtering is specified (that is, when `restrict` is specified on the first line of a statement), all routes from the specfied source will match that statement. If any filters are specified, only routes that match the specified filters will be exported. That is, if any filters are specified, a `all restrict ;` statement is assumed at the end of the list.

```
network [ exact | refines | between number and number ]
network mask mask [exact | refines | between number and number ] ]
network masklen number [ exact | refines | between number and number ] ]
default
host host
```

A.18.4.3 Specifying the Destination

As mentioned above, the syntax of the export statement varies depending on the protocol it is being applied to. One thing that applies in all cases is the specification of a metric. All protocols define a default metric to be used for routes being exported, in most cases this can be overridden at several levels of the export statement.

The specification of the source of the routing information being exported (the `export_list`) is described below.

Exporting to EGP and BGP

```
export proto bgp | egp as autonomous system
    restrict ;
export proto bgp | egp as autonomous system [ aspath-opt ]
    [ metric metric ] {
    export_list ;
} ;
```

Exportation to EGP and BGP is controlled by an autonomous system. The same policy is applied to all routers in the AS. EGP metrics range from 0 to 255 inclusive, with zero being the most attractive.

BGP metrics are 16 bit unsigned quantities; that is, they range from 0 to 65535 inclusive with 0 being the most attractive. While BGP version 4 actually supports 32 bit unsigned quantities, GATED does not yet support this. In BGP version 4, the metric is otherwise known as the Multi-Exit Discriminator, or MED.

In BGP, the `aspath-opt` option may be used to send the BGP `community` attribute. Any communities specified with the `aspath-opt` option are sent in addition to any received with the route or specified in the `group` statement.

If no export policy is specified, only routes to attached interfaces will be exported. If any policy is specified the defaults are overridden; it is necessary to explicitly specify everything that should be exported.

Note that EGP and BGP versions 2 and 3 only support the propagation of natural networks, so the host and default route filters are meaningless. BGP version 4 supports the propagation of any destination along with a contiguous network mask.

Exporting to RIP

```
export proto rip
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    restrict ;
export proto rip
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    [ metric metric ] {
    export_list ;
} ;
```

Exportation to RIP is controlled by any of protocol, interface or gateway. If more than one is specified, they are processed from most general (protocol) to most specific (gateway).

It is not possible to set metrics for exporting RIP routes into RIP. Attempts to do this are silently ignored.

If no export policy is specified, RIP and interface routes are exported into RIP. If any policy is specified, the defaults are overridden; it is necessary to explicitly specify everything that should be exported in the export_list.

When exporting routes from other protocols, it is important to specify a metric on the export statement or in the route filters. Unless this is done, the value specified in defaultmetric is used. If not specified, the defaultmetric value is 16 (unreachable). It is likely that this is not the desired result.

RIP version 1 assumes that all subnets of the shared network have the same subnet mask so they are only able to propagate subnets of that network. RIP version 2 removes that restriction and is capable of propagating all routes when not sending version 1 compatible updates.

To announce routes which specify a next hop of the loopback interface (that is, static and internally generated default routes) via RIP, it is necessary to specify the metric at some level in the export clause. Just setting a default metric for RIP is not sufficient. This is a safeguard to verify that the announcement is intended.

Exporting to OSPF

```
export proto osfpase [ type 1 | 2 ] [ tag ospf_tag ]
    restrict ;
export proto osfpase [ type 1 | 2 ] [ tag ospf_tag ]
    [ metric metric ] {
    export_list ;
} ;
```

It is not possible to create OSPF intra- or interarea routes by exporting routes from the GATED routing table into OSPF. It is only possible to export from the GATED routing table into OSPF ASE routes. It is also not possible to control the propagation of OSPF routes within the OSPF protocol.

There are two types of OSPF ASE routes, type 1 and type 2. The default type is specified by the defaults subclause of the ospf clause. This may be overridden by a specification on the export statement.

OSPF ASE routes also have the provision to carry a tag. This is an arbitrary 32 bit number that can be used on OSPF routers to filter routing information. The default tag specified by the OSPF defaults clause may be overridden by a tag specified on the export statement.

A.18.5 Specifying the Source

The export list specifies export based on the origin of a route and the syntax varies depending on the source.

Exporting BGP and EGP Routes

```
proto bgp | egp autonomoussystem autonomous_system
    restrict ;
proto bgp | egp autonomoussystem autonomous_system
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

BGP and EGP routes may be specified as the source autonomous system. All routes may be exported by AS path.

Exporting RIP Routes

```
proto rip
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    restrict ;
proto rip
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

RIP routes may be exported by protocol, source interface, or source gateway.

Exporting OSPF Routes

```
proto ospf | ospfase restrict ;
proto ospf | ospfase [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

Both OSPF, and OSPF ASE routes may be exported into other protocols.

Exporting Routes from Nonrouting Protocols

```
Non-routing with interface
```

```
proto direct | static | kernel
    [ (interface interface_list ) ]
    restrict ;
proto direct | static | kernel
    [ (interface interface_list ) ]
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

These protocols may be exported by protocol, or by the interface of the next hop. These protocols are:

- `direct` routes to directly attached interfaces.
- `static` static routes specified in a static clause.
- `kernel` on systems with the routing socket, routes learned from the routing socket are installed in the GATED routing table with a protocol of kernel. These routes may be exported by referencing this protocol. This is useful when it is desirable to have a script install routes with the ROUTE command and propagate them to other routing protocols.

Nonrouting by Protocol

```
proto default | aggregate
    restrict ;
proto default | aggregate
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

These protocols can only be referenced by protocol.

- default refers to routes created by the gendefault option. It is recommended that route generation be used instead.
- aggregate refers to routes synthesized from other routes when the aggregate and generate statements are used. See Section A.18.6 for more information.

Exporting by AS Path

```
proto proto | all aspath aspath_regexp
    origin any | ( [ igp ] [egp ] [ incomplete ] )
    restrict ;
proto proto | all aspath aspath_regexp
    origin any | ( [ igp ] [egp ] [ incomplete ] )
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

When BGP is configured, all routes are assigned an AS path when they are added to the routing table. For all interior routes, this AS path specifies IGP as the origin and no AS in the AS path; the current AS is added when the route is exported. For EGP routes, this AS path specifies EGP as the origin and the source AS as the AS path. For BGP routes, the AS path is stored as learned from BGP.

AS path regular expressions are described in Section A.18.2>

Exporting by Route Tag

```
proto proto | all tag tag restrict ;
proto proto | all tag tag
    [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
} ;
```

Both OSPF and RIP version 2 currently support tags, all other protocols always have a tag of zero. The source of exported routes may be selected based on this tag. This is useful when routes are classified by tag when they are exported into a given routing protocol.

A.18.6 Route Aggregation

Route aggregation is a method of generating a more general route given the presence of a specific route. It is used, for example, at an autonomous system border to generate a route to a network to be advertised using EGP, if one or more subnets of that network have been learned using RIP. Older versions of GATED automatically performed this function, generating an aggregate route to a natural network (using the old Class A, B and C concept), if there is an interface to a subnet of that natural network. However, that was not always the correct thing to do, and, with the advent of classless interdomain routing it is even more frequently the wrong thing to do. Therefore, aggregation must be explicitly configured. No aggregation is performed unless explicitly requested in an aggregate statement.

Route aggregation is also used by regional and national networks to reduce the amount of routing information passed around. With careful allocation of network addresses to clients, regional networks can just announce one route to regional networks instead of hundreds.

Aggregate routes are not actually used for packet forwarding by the originator of the aggregate route; they are used only by the receiver, if it wishes. A router receiving a packet that does not match one of the component routes that led to the generation of an aggregate route is supposed to respond with an ICMP network unreachable message. This is to prevent packets for unknown component routes from following a default route into another network where they would be forwarded back to the border router, and around and around again and again, until their TTL expires. Sending an unreachable message for a missing piece of an aggregate is only possible on systems with support for reject routes.

A slight variation of aggregation is the generation of a route based on the existence of certain conditions. This is sometimes known as the route of last resort. This route inherits the next hops and AS path from the contributor specified with the lowest (most favorable) preference. The most common usage for this is to generate a default based on the presence of a route from a peer on a neighboring backbone.

A.18.6.1 Aggregation and Generation Syntax

The syntax of the aggregate and generation statements are as follows:

```
aggregate default
    | ( network [ ( mask mask ) | ( masklen number ) ] )
    [ preference preference ] [ brief ] {
    proto [ all | direct | static | kernel | aggregate | proto ]
        [ ( as autonomous system ) | ( tag tag )
            | ( aspath aspath_regexp ) ]
        restrict ;
    proto [ all | direct | static | kernel | aggregate | proto ]
        [ ( as autonomous system ) | ( tag tag )
            | ( aspath aspath_regexp ) ]
        [ preference preference ] {
        route_filter [ restrict | ( preference preference ) ] ;
    } ;
} ;

generate dffault
| ( network [ ( mask mask ) | ( masklen )
    [ preference preference ] [ brief ] {
    proto [ all | direct | static | kernel | aggregate |
proto ]
        [ ( as autonomous system ) | ( tag tag
)
            | ( aspath aspath_regexp ) ]
        restrict ;
    proto [ all | direct | static | kernel | aggregate |
proto ]
        [ ( as autonomous system ) | ( tag tag
)
            | ( aspath aspath_regexp ) ]
        [ preference preference ] {
        route_filter [ restrict | ( preference preference ) ];
    } ;
} ;
```

Routes that match the route filters are called contributing routes. They are ordered according to the aggregation preference that applies to them. If there are more than one contributing routes with the same aggregating preference, the route's own preferences are used to order the routes. The preference of the aggregate route will be that of contributing route with the lowest aggregate preference.

- `preference` specifies the preference to assign to the resulting aggregate route. The default preference is 130.
- `brief` used to specify that the AS path should be truncated to the longest common AS path. The default is to build an AS path consisting of SETs and SEQUENCEs of all contributing AS paths.
- `proto` specifies, in addition to the special protocols listed, the contributing protocol may be chosen from among any of the ones supported (and currently configured into) GATED.
- `as` restricts selection of routes to those learned from the specified autonomous system.
- `tag` restricts selection of routes to those with the specified tag.
- `aspath` restricts selection of routes to those that match the specified AS path.
- `restrict` indicates that these routes are not to be considered as contributors of the specified aggregate. The specified protocol may be any of the protocols supported by GATED.

A route may only contribute to an aggregate route which is more general than itself; it must match the aggregate under its mask. Any given route may only contribute to one aggregate route, which will be the most specific configured, but an aggregate route may contribute to a more general aggregate.

Route Filters

All the formats allow route filters as shown below. See Section A.18.4.2 for a detailed explaination of how they work. When no route filtering is specified (that is, when restrict is specified on the first line of a statement), all routes from the specified source will match that statement. If any filters are specified, only routes that match the specified filters will be considered as contributors. That is, if any filters are specified, an `all restrict ;` statement is assumed at the end of the list.

```
network [exact | refines | between number and number ]
network mask mask [exact | refines | between number and number ]
network masklen number [ exact | refines | between number and number ] ]
default
host host
```

A.19 Sample Host Configurations

The configuration file for end systems is simple, usually containing only two configuration statements.

- The following sample configuration file emulates ROUTED. It runs RIP and only sends updates if there is more than one interface up and IP forwarding is enabled in the kernel:

```
#
rip yes ;
#
```

Note that RIP will not run if UDP checksums are disabled in the kernel.

- The following sample runs RIP in quiet mode; it only listens to packets, no matter how many interfaces are configured:

```
#
rip yes ;
{
   nobroadcast ;
} ;
#
```

- The following sample is suitable for any system that runs RIP and has only one network interface:

```
#
# do not time-out the network interface
#
interface 136.66.12.2 passive ;
#
# enable rip
#
rip yes ;
#
```

The `passive` keyword prevents GATED from changing the preference of the route to this interface if it is believed to be down due to lack of received routing information. The interface passive statement identifies a router with a guest host on an Ethernet.

In the example, the route is through the directly attached network interface. Normally, when GATED thinks an interface is down, it removes it from the routing database to prevent a gateway from announcing that it can route data through a nonoperational interface.

If the host has only one interface, it should not be removed from the routing database even if the interface is down (the interface 136.66.12.2 passive statement in the example). RIP is enabled with the `rip yes` statement. This statement is not required because it is the default, but the explicit statement in the GATED.CONF file serves to document the configuration to prevent future confusion.

A.19.1 Sample RIP and EGP Configuration

The following sample enables both an interior (RIP) and an exterior (EGP) protocol and sets certain protocol-specific parameters:

```
#  generate a default route if an EGP neighbor is acquired
#
options gendefault ;
#
# define the autonomous system number for EGP
#
autonomoussystem 303 ;
#
# enable RIP
#
rip yes ;
#
# enable EGP with hello interval 1 1/2 minute, poll
#  interval 10 minutes, neighbors 26.6.0.103 and 26.20.0.72
#
egp yes {
   packetsize 24488 ;
   group minhello 1:30 minpoll 10:00 {
       neighbor 26.6.0.103 ;
       neighbor 26.20.0.72 ;
   } ;
} ;
#
# announce 136.66 to AS 183
#
export proto egp as 183 {
   proto direct {
   136.66 metric 0 ;
   } ;
} ;
#
# announce default through  RIP with a metric of 3
#
export proto rip interface 136.66.12.1 {
 proto default {
 announce 0.0.0.0 metric 3 ;
 } ;
} ;
```

The AS number 303 is defined early because it is a definition statement and must occur before the first protocol statement. EGP is enabled by the yes keyword in the EGP statement. This statement also defines the following EGP parameters:

- `Packetsize` parameter, which defines the initial size of update packets accepted.
- `Group` clause, which sets parameters for all of the EGP neighbors in the group.
- `Minhello` and `minpoll`, which set the protocol timers.

The first `export` statement directs GATED to use EGP to advertise the network (136.66.0.0) to the Internet. This is the address of the network, not of a gateway. The second `export` statement is used to announce the default route to subnet 136.66.12.0 with a metric of 3.

A.19.2 Sample BGP and OSPF Configuration

The following sample implements the transformation of distance metrics between the internal (OSPF) and external (BGP) protocols. Autonomous system 1019, of which GATED is a member, contains network 19.0.0.0. The GATED machine has several interfaces into this autonomous system. The GATED daemon is using BGP to peer with AS 2021, neighbor 21.5.1.21.

```
# # # # # # # # # # # # # # # # # # #
interfaces {options all passive; };
autonomoussystem 1019;
routerid 19.1.1.18;
rip no;
hello no;
egp no;
bgp yes {
  preference 50 ;
  group type
  External peeras 2021
  {
    peer 21.5.1.21
    ;
  } ;
  group type
  IGP peeras 1019
  {
    peer 19.1.1.19
    ;
  } ;
} ;
ospf yes {
  area 0.0.0.2 {
    authtype none;
    networks {
      119.0.0.0 mask 255.0.0.0 ;
    } ;
    interface 119.2.128.18
    cost 1 {
       retransmitinterval 5;
       transitdelay 1;
       priority 1;
       hello interval 10;
       routerdeadinterval 40;
    } ;
    interface 119.4.128.18
    cost 1 {
       retransmitinterval 5;
       transitdelay 1;
       priority 1;
       hellointerval 60;
       routerdeadinterval 180;
    } ;
  } ;
```

```
  backbone {
    authype none;
    interface 19.1.1.19
    cost 1 {
       retransmitinterval 5;
       transitdelay 1;
       priority 1;
       hellointerval 60;
       routerdeadinterval 180;
    } ;
  } ;
} ;
export proto ospfase type 1 {
  proto bgp as 2021 {
    ALL
    metric 1; };
  proto direct {
    ALL
    metric 1;  };
} ;
export proto bgp as 2021 {
  proto direct {
    ALL
    metric 1; } ;
  proto ospfase {
    ALL
    metric 1; } ;
} ;
```

In this example, two autonomous systems (one internal, one external) are directly connected through a router that is attached to a backbone speaking OSPF. The AS number 1019 is defined early, because it is a definition statement that occurs again in the first protocol statement, which enables BGP. The first export statement directs GATED to advertise routes from the internal group AS 1019. The group AS 1019 is running OSPF as its interior gateway protocol and is running BGP as its exterior routing protocol to route information to the external group AS 2021.

Routes to two local Ethernets in AS 1019, identified as 119.2.128.18 and 119.4.128.18 (119.0.0.0 mask 255.0.0.0), are advertised along with the OSPF backbone (19.1.1.19). The parameters for AS path, path origin, and transitive optional attributes, including transmission intervals, are defined. The second export statement announces the default route to AS 2021 with a metric of 1.

A.20 For More Information

For more information about configuring GATED routing, visit the GATED Consortium web page:

www.gated.org

B

EBCDIC/DMCS Translation Tables

The TCP/IP Services TELNET implementation supports IBM 3270 terminal emulation. The default translation tables satisfy most users' needs.

B.1 Macros for Modifying the Translation Tables

If the standard translation table does not suit your needs, you can modify it by specifying macros in the file TN3270DEF.MAR. You should copy TN3270DEF.MAR from TCPIP$EXAMPLES into your current default directory and edit it with any editor supported by your system.

Use the macros described to make any changes you need in the translation tables. You can specify three macros. The arguments for all three macros are:

eb The EBCDIC code for the character you want to translate.

as The DMCS code for the character you are translating to. (You can specify the actual DMCS display character instead of the code, if you want to. To do this, enter a single quotation mark before you type the character, for example, '!, 'A, 'g, and so on.)

The macros are:

- EB2AS *eb, as*

  The EB2AS macro lets you change an entry in the EBCDIC-to-DMCS table without affecting the DMCS-to-EBCDIC table. For example:

  EB2AS 5A, '!

  In this example, the EBCDIC hexadecimal code 5A is translated to the DMCS exclamation point (hexadecimal code 21). The macro does not affect the translation of a DMCS exclamation point to its EBCDIC equivalent.

- AS2EB *as, eb*

  The AS2EB macro lets you change an entry in the DMCS-to-EBCDIC table without affecting the EBCDIC-DMCS table. For example:

  AS2EB '[, 5F

  In this example, the DMCS open bracket character (hexadecimal code 5B) is translated to the EBCDIC hexadecimal code 5F. The macro does not affect the translation of the EBCDIC code 5F to DMCS.

- REVTRA *eb, as*

  The REVTRA macro combines the functions of the EB2AS and AS2EB macros, enabling you to change the same translation in both the DMCS-to-EBCDIC and EBCDIC-to-DMCS tables. For example:

  REVTRA 4A, A2

In this example, the macro changes the EBCDIC-to-DMCS translation table so that the EBCDIC character represented by the hexadecimal code 4A translates to a DMCS cent sign (hexadecimal code A2.) The DMCS-to-EBCDIC translation table is also changed so that a DMCS cent sign translates to the EBCDIC character represented by the hexadecimal code 4A.

NOTE

If you use the REVTRA macro, you must give new translations to the codes used as arguments to the macro. You can do this with the EB2AS and AS2EB macros.

B.2 Building Translation Tables

Before you edit the file TN3270DEF.MAR, save the original by copying it from TCPIP$EXAMPLES to your current default directory. Edit the file in your own directory.

Edit the file using any editor your system supports. When you have changed the file to your satisfaction, perform the following steps:

1. Assemble the file you just edited:

   ```
   $ MACRO/OBJECT TN3270DEF
   ```

   When you assemble the template file, you create an object file containing two 256-byte translation tables labeled $AS2EB:: and $EB2AS::. This object file can be linked to a user application program.

2. Link the new file to create the translation table, enter:

   ```
   $ LINK/SYSTEM/HEADER TN3270DEF
   ```

3. Copy the resulting image to the system library. Enter:

   ```
   $ COPY TN3270DEF.EXE SYS$LIBRARY:TN3270DEF.TBL
   ```

   The .EXE file is renamed to .TBL in this final step.

B.3 Examples of Modifying Translation Tables

This section gives two examples of modifying translation tables. Example 1 shows how to translate the ASCII left bracket to the EBCDIC cent sign. Example 2 shows how to modify the standard translation tables to the translation tables used by the TN3270 Terminal Emulator.

1. The following code segment translates the ASCII left bracket, hexadecimal code 5B, to the EBCDIC cent sign, hexadecimal code 4A. The change causes the EBCDIC cent sign to be translated into the ASCII cent sign, hexadecimal A2. When the REVTRA macro is used, it leaves the ASCII left bracket unmapped, and a second macro, AS2EB, is used to map the ASCII left bracket to the EBCDIC SUB character, hexadecimal 3F.

   ```
   DMFILL  = 26.           ; This argument causes all the EBCDIC
                           ; characters that normally map to an ASCII
                           ; backslash in the standard table to map
                           ; to an ASCII SUB character, code 26
                           ; decimal, 1A hexadecimal.
   ```

```
REVTRA  4A,A2           ; Map the EBCDIC cent character (4A)
                        ; to/from the ASCII cent character (A2).
AS2EB   5B,3F           ; Map the ASCII "[" (5B) to the EBCDIC
                        ; SUB character (3F).
```

The preceding macro could also be written in the following way:

```
AS2EB '[,3F
```

2. The following example shows the macros used to modify the standard translation tables to the translation tables used by IBM 3270TE.

```
DMFILL  = 26.
REVTRA  4A,A2           ; Map the EBCDIC cent character (4A)
                        ; to the ASCII cent character (A2).
                        ; Because this macro leaves ASCII "[" (5B)
                        ; still mapped to the EBCDIC cent character
                        ; (4A), it must be remapped.
REVTRA  4F,7C           ; Map the EBCDIC "|" (4F) to/from
                        ; the ASCII "|" (7C).
REVTRA  6A,A1           ; Map EBCDIC "dashed vbar" (6A) to/from ASCII
                        ; inverted ! (A1).
REVTRA  5A,'!           ; Map EBCDIC "!" (5A) to/from ASCII "!" (21).
AS2EB   ']',3F          ; Map ASCII "]" (5D) to the EBCDIC SUB
                        ; character (3F).
AS2EB   5B,3F           ; Map the ASCII "[" (5B) to the EBCDIC
                        ; SUB character (3F).
```

The changes that are described modify a version of the ANSI standard X3.26 1970 EBCDIC-to-ASCII translation table. Table B–1 shows these modifications:

Table B–1 Modifications to Translation Tables

DMCS Character	Hexadecimal Code	EBCDIC Character	Hexadecimal Code
¢	A2	¢	4A
|	7C	|	4F
!	21	!	5A
¡[1]	A1	dashed vbar	6A
[	5B	[2]	
]	5D	[2]	

[1]The display of these characters depends on the type of terminal.

[2]These characters translate to the EBCDIC SUB character, which has an EBCDIC code of 63 decimal (3F hexadecimal).

The DMCS contains 256 characters. The first 128 characters are the same as the standard ASCII character set. None of the remaining characters map to a printable EBCDIC character; therefore, they translate to the EBCDIC SUB character.

C

How NFS Converts File Names

The NFS to OpenVMS file name translation rules in Table C–1 are based on the character mapping scheme in Table C–2. The OpenVMS to NFS mapping rules are the converse of these rules.

Table C–1 NFS Server to OpenVMS Client File Name Conversion Rules

Rule	What Happens to File Names from NFS to OpenVMS
1	Lowercase characters become uppercase (unless Rule 2 applies). For example, `file` becomes FILE.;1
2	Initial uppercase characters or a sequence of case-shifted characters are prefixed with the "$" escape character. For example, `CaseShiftedFile` becomes CASESHIFTEDFILE.;1
3	A file without a version gets a version number preceded by a semicolon. For example, `file` becomes FILE.;1
4	If a file name does not include a dot (.), a dot is added before the version number semicolon. For example, `file` becomes FILE.;1
5	After its name is converted, a file will not appear in an OpenVMS directory listing if any one of the following criteria are met: • The file name is more than 39 characters long. • The file extension is more than 39 characters long. • The version number is greater than 32767.
6	If the file name has a dot, the dot is preserved unless the resulting file name fails one of the tests in Rule 5; if so, the dot becomes "$5N" and the same rule applies to each subsequent dot found. For example, `more.file.text` becomes MORE.FILE$5NTEXT;1
7	If the file name is a directory, each dot becomes "$5N" and the file name gets the ".DIR" extension. For example, `dot.directory.list` becomes DOT$5NDIRECTORY$5NLIST.DIR;1
8	Invalid OpenVMS characters become the escape character sequences in the second column of Table C–2 ("$" followed by a digit and a letter). For example, `special#character&file` becomes SPECIAL$5CCHARACTER$5FFILE.;1 ("#" becomes "$5C" and "&" becomes "$5F")
9	Any existing "$" becomes "$$" (plus any "$" added due to Rule 2 or 8 above). For example, `dollar$Sign$5cfile` becomes DOLLAR$$$S$IGN$$5CFILE.;1

Table C–2 provides a complete list of OpenVMS character sequences, corresponding server characters, and octal values used for NFS name conversion.

Table C–2 NFS Client Name Conversion

OpenVMS Character Sequence	Server Character	Octal Value
$6A	<CTRL/@>	000
$4A	<CTRL/A>	001
$4B	<CTRL/B>	002
$4C	<CTRL/C>	003
$4D	<CTRL/D>	004
$4E	<CTRL/E>	005
$4F	<CTRL/F>	006
$4G	<CTRL/G>	007
$4H	<CTRL/H>	010
$4I	<CTRL/I>	011
$4J	<CTRL/J>	012
$4K	<CTRL/K>	013
$4L	<CTRL/L>	014
$4M	<CTRL/M>	015
$4N	<CTRL/N>	016
$4O	<CTRL/O>	017
$4P	<CTRL/P>	020
$4Q	<CTRL/Q>	021
$4R	<CTRL/R>	022
$4S	<CTRL/S>	023
$4T	<CTRL/T>	024
$4U	<CTRL/U>	025
$4V	<CTRL/V>	026
$4X	<CTRL/W>	027
$4X	<CTRL/X>	030
$4Y	<CTRL/Y>	031
$4Z	<CTRL/Z>	032
$6B	<CTRL/[>	033
$6C	<CTRL/\ >>	034
$6D	<CTRL/]>	035
$6E	<CTRL/^>	036
$6F	<CTRL/_>	037
$7A	<SPACE>	040
$5A	!	041
$5B	"	042
$5C	#	043
$5E	%	045

(continued on next page)

Table C–2 (Cont.) NFS Client Name Conversion

OpenVMS Character Sequence	Server Character	Octal Value
$5F	&	046
$5G	’	047
$5H	(	050
$5I	)	051
$5J	*	052
$5K	+	053
$5L	,	054
$5N	.	056
i$5O	/	057
$5Z	:	072
$7B	;	073
$7C	<	074
$7D	=	075
$7E	>	076
$7F	?	077
$8A	@	100
$8B	[	133
$8C	\	134
$8D	]	135
$8E	^	136
$9A	‘	140
$9B	{	172
$9C	|	174
$9D	}	175
$9E	~	176
$9F	<DEL>	177

D
Acronyms

Table D–1 shows DIGITAL TCP/IP Services for OpenVMS acronyms and other acronyms related to TCP/IP networking.

Table D–1 Acronyms

Acronym	Meaning
ACK	acknowledgment
ACL	access control list
ACP	ancillary control process
ANSI	American National Standards Institute
API	application programming interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ATM	asynchronous transfer mode
BBS	Bulletin Board System
BGP	Border Gateway Protocol
BIND	Berkeley Internet Name Domain
BOOTP	Bootstrap Protocol
bps	bits per second
BSD	Berkeley Software Distribution
CFS	container file system
CFSRTL	container file system run-time library
CSLIP	Compressed Serial Line Internet Protocol
DCE	Distributed Computing Environment
DCL	Digital Command Language
DEK	data encryption key
DES	data encryption standard
DNS	Domain Name Service
eSNMP	extensible Simple Network Management Protocol
EGP	External Gateway Protocol
FDDI	Fiber Distributed Data Interface
EOF	end of file
EOL	end of line
FQDN	fully qualified domain name

(continued on next page)

Table D–1 (Cont.) Acronyms

Acronym	Meaning
FTP	File Transfer Protocol
GID	group identification (UNIX)
IAB	Internet Architecture Board
ICMP	Internet Control Message Protocol
IGP	Internal Gateway Protocol
InterNIC	Internet Network Information Center
IP	Internet Protocol
ISDN	Integrated Services Digital Networks
IVP	installation verification procedure
Kbps	kilobits per second
LAN	local area network
LPD	line printer daemon
LPR	remote line printing
MBUF	memory buffer
MFD	master file directory
MIB	Management Information Base
MIB II	Management Information Base II
MTU	maximum transmission unit
MX	mail exchange
NAK	negative acknowledgment
NFS	Network File System
NIS	Network Information Service
NMS	Network Management Station
NOC	Network Operations Center
NTP	Network Time Protocol
PDU	protocol data unit
PING	packet internet groper
POP	Post Office Protocol
PPP	Point-to-Point Protocol
PSDN	Packet Switching Data Network
PWIP	PATHWORKS Internet Protocol
RARP	Reverse Address Resolution Protocol
RCP	remote copy
REXEC	remote execute
RFC	Request for Comments
RLOGIN	remote login
RIP	Routing Information Protocol
RMS	Record Management Services

(continued on next page)

Table D–1 (Cont.) Acronyms

Acronym	Meaning
RPC	remote procedure call
RSH	remote shell
RTL	run-time library
RTT	round-trip time
SLIP	Serial Line Internet Protocol
SMI	structure of management information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TAC	terminal access controller
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial File Transport Protocol
TP	Time Protocol
TTL	time to live
UAF	user authorization file
UCP	Management Control Program
UDP	User Datagram Protocol
UID	user identification (UNIX)
UTC	Universal Coordinated Time
WAN	wide area network
WKS	well known service
XDR	external data representation

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

X

Z