TCPware® for OpenVMS Management Guide

Part Number: N-5200-02-NN-A

September 1999

This manual provides the system manager with the procedures for managing the TCPware for OpenVMS family of software products.

Revision/Update: This is a revised manual.

Software Version: 5.4

Process Software Corporation
Framingham, Massachusetts
USA

Contents

Chapter 2 DHCP/BOOTP Server

Chapter 3 Common Interfaces

Chapter 4 Serial Link Interfaces: PPP and SLIP

Chapter 7 X.25 Interface

Chapter 8 Routing and GateD

Chapter 9 Network Time Protocol (NTP)

Chapter 10 TIMED

Chapter 11 Managing FTP-OpenVMS

Chapter 17 Managing TELNET-OpenVMS Server

Chapter 18 Managing TCPware Security

Chapter 19 Access Restrictions

Chapter 20 Packet Filtering

Chapter 21 Managing Token Authentication

Chapter 22 Managing Kerberos

Chapter 23 IP Security Option

Chapter 24 PATHWORKS Support

Chapter 25 Tunneling DECnet over IP

NFS-to-OpenVMS Filename Mapping

Data Network Identification Codes

Preface

Introducing This Guide

This guide describes how to manage a TCP/IP network and the TCPware components. It is for system managers and administrators.

What You Need to Know Beforehand

Before using TCPware, you should be familiar with:

- The TCPware for OpenVMS products, components, features, and capabilities (see the *User's Guide* for more information)
- Computer networks in general
- Compaq Computer Corporation's OpenVMS operating system and file system

How This Guide Is Organized

This guide has the following contents:

- Part I, *Managing Hosts* – Includes chapters on the Domain Name Services, and the Dynamic Host Configuration Protocol (DHCP) and BOOTP servers.
- Part II, *Managing Networks* – Includes chapters on network interfaces (Ethernet, FDDI, POP3, and so on), Cluster Alias Failover, the Simple Network Management Protocol (SNMP), and X.25 networks.
- Part III, *Managing Routing* – Includes a chapter on the routing protocols (primarily GateD).
- Part IV, *Managing Time Services* – Includes chapters on the Network Time Protocol (NTP) and TIMED protocol.
- Part V, *Managing Applications* – Includes chapters on managing the FTP-OpenVMS Client and Server, NFS-OpenVMS Client and Server, Print Services, Berkeley R Command services, the mail services (SMTP-OpenVMS and IMAP), and TELNET-OpenVMS Server.

- Part VI, *Managing Security* – Includes chapters on general TCPware security, Access Restrictions, Packet Filtering, Token Authentication, Kerberos server and applications, and the IP Security Option (IPSO).
- Part VII, *Managing Additional Support* – Includes chapters on PATHWORKS support, tunneling DECnet over IP, X Display Manager (XDM), and DECwindows support.
- Part VIII, *Network Testing Tools* – Includes a chapter on the network testing tools, such as DISCARD, FINGER, NETCU DEBUG, NSLOOKUP, PING, QUOTED, TCPDUMP, and TRACEROUTE.
- Appendixes, including NFS-to-OpenVMS filename mapping rules, and Data Network Identification Codes for X.25 networks.
- Index to this guide.

Online Help

You can use help at the DCL prompt to find the following:

- Topical help – Access TCPware help topics only as follows:

  `$ HELP TCPWARE [topic]`

  The topic entry is optional. You can also enter topics and subtopics at the following prompt and its subprompts:

  `TCPWARE Subtopic?`

  Online help is also available from within certain TCPware components: FTP-OpenVMS Client and Server, Network Control Utility (NETCU), TELNET-OpenVMS Client, NSLOOKUP, and TRACEROUTE. Use the HELP command from within each component.

  Example: `NETCU> HELP [topic]`

- Error messages help – Access help for TCPware error messages only as follows:

  `$ HELP TCPWARE MESSAGES`

  If the error message is included in the MESSAGES help, it identifies the TCPware component and provides a meaning and user action. See the `Instructions` under `MESSAGES`.

Obtaining Customer Support

You can use the following customer support services for information and help about TCPware and other Process Software products if you subscribe to our Product Support Services. (If you bought TCPware products through an authorized TCPware reseller, contact your reseller for technical support.) Contact Technical Support directly using the following methods:

- **Electronic Mail**

  E-mail relays your question to us quickly and allows us to respond, as soon as we have information for you. Send e-mail to **support@process.com**. Be sure to include your:

- Name
- Telephone number
- Company name
- Process Software product name and version number
- Operating system name and version number

Describe the problem in as much detail as possible. You should receive an immediate automated response telling you that your call was logged.

- **Telephone**

  If calling within the continental United States or Canada, call Process Software Corporation Technical Support toll-free at 1-800-394-8700. If calling from outside the continental United States or Canada, dial 1-508-628-5074. Provide your name, company name, and telephone number. If no one is available to take your call immediately, a technical support representative will return your call as soon as possible.

- **FAX**

  Sending a FAX to 508-879-0042 is another way to log a technical support call. Be sure the FAX includes all the information listed under "Electronic Mail" plus your FAX number as reference. Describe the problem in as much detail as possible. A diagram of your network is often very useful.

- **World Wide Web**

  There is a variety of useful technical information available on our World Wide Web home page, **`http://www.process.com`** (select **`Customer Support`**).

- **Internet Newsgroup**

  You can also access the VMSnet newsgroup, **`vmsnet.networks.tcp-ip.tcpware`**. Process Software's Engineering and Technical Support professionals monitor and respond to this open forum newsgroup on a timely basis.

Licensing Information

TCPware for OpenVMS includes a software license that entitles you to install and use it on one machine. Please read and understand the *Software License Agreement* before installing the product. If you want to use TCPware on more than one machine, you need to purchase additional licenses. Contact Process Software or your distributor for details.

Maintenance Services

Process Software offers a variety of software maintenance and support services. Contact us or your distributor for details about these services.

Reader’s Comments Page

TCPware guides may include Reader’s Comments as their last page. If you find an error in this guide or have any other comments about it, please let us know. Return a completed copy of the Reader’s Comments page, or send e-mail to **`techpubs@process.com`**.

Please make your comments specific, including page references whenever possible. We would appreciate your comments about our documentation.

Documentation Set

The documentation set for TCPware for OpenVMS consists of the following:

- ***Release Notes*** for the current version of TCPware for OpenVMS **–** For all users, system managers, and application programmers. The *Release Notes* are available online on your TCPware for OpenVMS media and are accessible before or after software installation.
- ***Installation & Configuration Guide*** – For system managers and those installing the software. The guide provides installation and configuration instructions for the TCPware for OpenVMS products.
- ***User's Guide*****–** For all users. This guide includes an introduction to TCPware for OpenVMS products as well as a reference for the user functions arranged alphabetically by product, utility, or service.
- ***Management Guide*** – For system managers. This guide contains information on functions not normally available to the general network end user. It also includes implementation notes and troubleshooting information.
- ***Network Control Utility (NETCU) Command Reference*** – For users and system managers. This reference covers all the commands available with the Network Control Utility (NETCU) and contains troubleshooting information.
- ***Programmer’s Guide*** – For network application programmers. This guide gives application programmers information on the callable interfaces between TCPware for OpenVMS and application programs.
- Installation Quick Reference Card
- Network Control Utility (NETCU) Quick Reference Card
- **Online help –**
  - Topical help, using **`HELP TCPWARE`** ***`[topic]`***
  - Error messages help, using **`HELP TCPWARE MESSAGES`**

Conventions Used

Convention	Meaning
host	Any computer system on the network. The local host is your computer. A remote host is any other computer.
monospaced type	System output or user input. User input is in `bold type`. Example: `Is this configuration correct? YES` Monospaced type also indicates user input where the case of the entry should be preserved.
italic type	Variable value in commands and examples. For example, *username* indicates that you must substitute your actual username. Italic text also identifies documentation references.
[*directory*]	Directory name in an OpenVMS file specification. Include the brackets in the specification.
[optional-text]	(Italicized text and square brackets) Enclosed information is optional. Do not include the brackets when entering the information. Example: `START/IP line` *`address [info]`* This command indicates that the *info* parameter is optional.
{*value* | *value*}	Denotes that you should use only one of the given values. Do not include the braces or vertical bars when entering the value.
Note!	Information that follows is particularly noteworthy.
CAUTION!	Information that follows is critical in preventing a system interruption or security breach.
$\boxed{\textit{key}}$	Press the specified key on your keyboard.
$\boxed{\textit{Ctrl/key}}$	Press the control key and the other specified key simultaneously.
$\boxed{\texttt{Return}}$	Press the Return or Enter key on your keyboard.
——►	Pointer to further reading or a cross-reference.

PART I Managing Hosts

Chapter 1

Domain Name Services

Introduction

This chapter describes TCPware's Domain Names Services and how it relates to the Domain Name System (DNS). Specific sections describe:

- The different types of DNS servers, the DNS client, and what they do.
- The syntax, special commands, and fields used in each type of database record.
- TCP/IP cluster load balancing.

Domain Name Services allows local hosts to obtain information about other hosts by accessing a distributed DNS database. This database supplies Internet addresses and hostnames throughout the network. Any upper-layer protocol, such as FTP or SMTP, can use DNS when it needs host information.

Domain Name Services also provides the services for cluster load balancing.

Domain Name System (DNS) Concepts

For a full description of the concepts behind the Domain Name System (DNS) that forms the basis of the Domain Name Services, see *DNS Defined: A Practical Guide to TCP/IP Domain Name System and Services. DNS Defined* is one of the volumes in Process Software Corporation's useful *TCP/IP Reference Library.*

DNS Client

A DNS client (also called a resolver) communicates with a DNS server to resolve a host name and internet address. The client does not maintain a database. The client only sends queries and provides the received answers to applications.

You can configure your local host to support a DNS client when you run the TCPware configuration. The host can support a DNS client only, or both a client and name server.

The configuration procedure (CNFNET) prompts you to specify the internet addresses of up to three name servers the client can query. The client reads this information from two logicals you set through CNFNET:

- TCPWARE_DOMAINNAME
- TCPWARE_NAMESERVERS

The client also allows you to set up to six domains in a search list, as well as the minimum number of dots to recognize in a host name to make it fully qualified. The client reads this information from two logicals you set through CNFNET:

- TCPWARE_DOMAINLIST
- TCPWARE_ RES_OPTIONS "ndots: *ndots*"

See the *Installation & Configuration Guide,* Chapter 4, *Configuring the TCP/IP Services,* the *Configure the DNS Resolver* section.

When an application needs to resolve a host name or internet address, the client queries the first name server the TCPWARE_NAMESERVERS logical defines. The client continues to query the other name servers on its list until it receives an answer or the list is exhausted. The amount of time varies for what is needed to send a query and receive a response.

The TCPware resolver queries domain names in the following sequence:

1	The resolver queries the fully qualified domain name if the domain name contains *ndots* number of dots (see the TCPWARE_RES_OPTIONS logical syntax); the default is `1`.
2	If the first step fails or the domain name contains less than *ndots*, the resolver queries *name*.*default-domain* or the list established by the TCPWARE_DOMAINLIST logical.
3	If the second step fails, the resolver queries for *name* only.

You can restart the resolver process (TCPware_DNS) if it goes down (or the NETCU STOP/DNS command was used) by specifying **`@TCPWARE:STARTUP_RESOLVER DETACH`** on the command line.

Domain Name Server

To set up a domain name server on your local host, perform these steps:

1	Determine whether the names on the local host will be authoritative for a zone or will be just a caching server.

2	Run the CNFNET configuration procedure described in the *Installation & Configuration Guide*, Chapter 4. When you set up a server and (there is no existing NAMED.CONF file), CNFNET converts the existing TCPWARE_NAMED_ROOT:NAMED.BOOT file to a TCPWARE_NAMED_ROOT:NAMED.CONF file. If there is no NAMED.BOOT file, CNFNET creates a default NAMED.CONF file set up as a caching server.
3	If the host is to be authoritative, add zone statements in the NAMED.CONF file (see Zone) for each zone.
4	Gather the information you need for the database files so that the server can resolve queries. The information is in the *Resource Records* section of this chapter.
5	Enter the information in the database files using a text editor. (See *Database Files*.)

Starting and Stopping

The name server starts up when you execute the STARTNET.COM procedure and shuts down with the SHUTNET.COM procedure.

1	Log in as the system manager.
2	Stop the name server process: `$` **`@TCPWARE:SHUTNET DNS`**
3	Start the name server process: `$` **`@TCPWARE:STARTNET DNS`**

When you start the name server, it creates a log file, TCPWARE:NAMESERVER.LOG. This file contains information about any problems uncovered when loading the database or during name server operation.

If the name server is already active and you edited the database files, you can restart the name server or you can reload it using the NETCU RELOAD NAMED command. See Chapter 2 of the *Network Control Utility (NETCU) Command Reference*.

You can investigate this file after updating the database or if the name server exits immediately after startup. Syntax errors in the database files are the most common source of errors. If there is an error in one of the database files, Domain Name Services records the error type and relevant database file in the log file.

Database Files can help you find the cause of these errors.

The TCPWARE_NAMED_ROOT:NAMED.CONF File

The main DNS configuration file, from which the name server gets its initial data, is TCPWARE_NAMED_ROOT:NAMED.CONF. The equivalent of this file in UNIX-based BIND implementations is /etc/named.conf. Use this file to add information about your site when setting up a master DNS server. An example configuration file follows.

```
/*
** Sample Configuration File for DNS server
```

```
*/

options {
    directory "TCPWARE_ROOT:[TCPWARE.NAMED]";
//  forward only;
    forwarders { 128.0.1.1; 128.0.2.10; };
};

zone "flowers.com" in {
    type master;
    file "domain-name-service.iris";
};

zone "0.128.in-addr.arpa" in {
    type master;
    file "domain-name-service.iris-net";
};

zone "cc.flowers.com" in {
    type slave;
    masters { 128.0.1.1; };
    file "domain-name-service.cc";
};

zone "1.0.128.in-addr.arpa" in {
    type slave;
    masters { 128.0.1.1; };
    file "domain-name-service.cc-net";
};

zone "0.0.127.in-addr.arpa" in {
    type master;
    file "domain-name-service.local";
};

zone "." in {
    type hint;
    file "domain-name-service.cache";
};
```

The following sections describe the zone, options, and logging sections.

Zone

A *zone* is that part of a name server that contains complete information about the domain name space. You specify a *zone* is the following way:

```
zone “<domain name>” [<class>] {
            type <type>;
        };
```

Table 1-1 defines the NAMED.CONF zone fields.

Table 1-1 NAMED.CONF Zone Fields

Field	Description
<class>	The class to which this zone applies. If the class is not specified, the type IN is used by default. The syntax is **[(in | hs | hesiod | chaos)]**, • in (default)--Used for objects connected to the Internet. This is the only supported type. • hs or hesiod--Confined mostly to MIT. hs is the abbreviation for hesiod. • chaos--An historic network. Not used today.
“<domain name>”	The name of the domain for which this zone is authoritative.
file "<filename>";	Specifies the name of the file.
masters { ip_addr; [ip_addr; ...]};	Specifies the IP addresses from where the server is to transfer the zone data. This statement is meaningful only for slave or stub zones.
type (master | slave | stub | hint);	See Table 1-3 for a description of these zones.

Table 1-2 Optional Zone Statements

Statement	Description
allow-query { <address_match_list> }; allow-transfer { <address_match_list> };	Overrides the respective statement in the global options section for this zone. See Table 1-5.
allow-update { <address_match_list> };	Specifies the addresses of hosts that are allowed to modify the zone with dynamic updates. Defaults to none.
also-notify { ip_addr; [ip_addr; ...] };	Lists the servers to send zone change notifications to as well as to the slave servers specified via the NS records for the zone.
check-names (warn | fail | ignore);	Overrides the default name checking specified in the global options section. See the *check-names* statement in Table 1-4 for more details.

Table 1-2 Optional Zone Statements (Continued)

Statement	Description
notify (yes | no);	Specifies if zone change notifications should be sent to the slave servers for the zone. This overrides the *notify* statement in the global options section. See the *notify* statement in Table 1-4 for more details.

Table 1-3 Zone Types

Type	Description
hint	Specifies that data in the DOMAIN-NAME-SERVICE.CACHE file, which is in standard resource record format, should be placed in the bootstrap cache. The hint zone definition is used to specify locations of root domain servers. An up-to-date list of root name servers is automatically obtained and stored in memory without replacing the cache file.
master	Specifies data for the zone and the domain. The first master zone definition states that the file DOMAIN-NAME-SERVICE.IRIS contains authoritative data for the FLOWERS.COM zone, in standard resource record format. The second master zone definition states that the file DOMAIN-NAME-SERVICE.IRIS-NET contains authoritative data for the domain 0.128.IN-ARPA.ARPA, which is used in translating addresses in network 128.0.0.0 to host names. Be sure each zone master file begins with an SOA (Start Of Authority) resource record for the zone, as shown in the *DNS Zone Information Files* section.

Table 1-3 Zone Types (Continued)

Type	Description
slave	Specifies the zones for which this DNS server acts as a secondary name server. After this name server receives a "zone transfer," it becomes authoritative for the specified zone. The first slave zone definition specifies that all authoritative data under CC.FLOWERS.COM is to be transferred from the name server at 128.0.1.1. The file statement in this section is the file name in which to back up the transferred zone. When it boots, the name server loads the zone from this backup file, if it exists, providing a complete copy even if the master DNS server is unreachable. This file is updated whenever a new copy of the domain is received by automatic zone transfer from one of the master servers. The file statement is optional, but recommended to speed up server startup and eliminates needless bandwidth. The second slave zone definition states that the address-to-hostname mapping for the subnet 128.0.1.0 should be obtained from the same list of master servers as the previous zone.
stub	Works like a slave zone, except it transfers only the nameserver records for the master zone rather than the full zone information.

Options

The *options* statement sets up global options to be used by NAMED. Use this statement only once in a configuration file. If it is used more than once, the first occurrence determines what options to use, and a warning is generated. If no *options* statement is present, an options block is used setting each option to its default value.

You specify *options* in the following way:

```
options {
            <options statements>
         };
```

Table 1-4 defines the NAMED.CONF options.

Table 1-4 NAMED.CONF Options

Option	Description
allow-query { address_match_list };	See the Address_match_list section. Specifies the addresses of hosts that are allowed to query the server for information. It defaults to all.

Table 1-4 NAMED.CONF Options (Continued)

Option	Description
allow-transfer { address_match_list };	See the Address_match_list section. Specifies the addresses of hosts that are allowed to perform zone transfers from the server. It defaults to all.

check-names (master | slave | response) (warn | fail | ignore);	The server checks names in three areas:
	• Master zone files.
	• Slave zone files.
	• Responses to queries the server has initiated.
	
	The server assumes the following defaults:
	
	`options {`
	`    check-names master fail;`
	`    check-names slave warn;`
	`    check-names response ignore;`
	`};`
	
	• 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.
	
	If *check-names response fail* has been specified, and answering the client's question requires sending an invalid name to the client, the server sends a REFUSED response code to the client.

directory "<path>";	Causes the server to change its default directory to the specified directory. This can be important for the correct processing of $INCLUDE files in primary zone files, or file statements in zone definitions.
fake-iquery (yes | no);	If *yes*, the server responds to the obsolete IQUERY query type. The default is *no*.

Table 1-4 NAMED.CONF Options (Continued)

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>fetch-glue (yes | no);</td><td>If yes (default), the server queries for the resource records it needs to complete the additional section of a response.

To prevent the server’s cache from growing, use fetch-glue no; in combination with recursion no;.</td></tr>
<tr><td>forward (only | first);</td><td>This statement is meaningful only if there is a forwarders statement.

When first (default) is used, the server queries the forwarders first before consulting the root domain servers.

When only is used, the server queries the forwarders only. If the forwarders fail to find an answer, the server does not query the root domain servers.

<pre>options {
  forward only;
  forwarders
         { 192.1.1.98; 192.1.1.99;};
};</pre></td></tr>
<tr><td>forwarders { ip_addr; [ip_addr; ...] };</td><td>Specifies the addresses of site-wide servers that accept recursive queries from other servers. If the DNS server configuration file specifies one or more forwarders, the server sends all queries for data not in the cache to the forwarders.

Central name servers designated to handle forwarded requests can then develop a cache of answers to external queries. The central cache reduces the number of requests sent to root name servers and improves DNS performance.</td></tr>
</table>

Table 1-4 NAMED.CONF Options (Continued)

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>listen-on [port ip_port]
{ address_match_list };</td><td>See the Address_match_list section.

Specifies what port on what interface to listen on. The default is: listen-on port 53 { any };

Example:<pre>
options {
   // listen on port 53 for
   // external interfaces.
   listen-on { 192.42.95.0;};
   // listen on port 43 for
   // internal interfaces.
   listen-on port 43
     { 127.0.0.1; 10.0.0.0; };
};
</pre></td></tr>
<tr><td>notify (yes | no);</td><td>If yes (default), the server notifies slave servers if there are any changes to a domain for which the server is master. The server determines the slave servers by the nameserver records contained in the zones data file.

For more information, see the also-notify statement in Table 1-4.</td></tr>
<tr><td>recursion (yes | no);</td><td>If yes (default), the server attempts to do all the work required to answer a query that has requested recursion. Turning this off results in the server responding to the client with referrals.

To prevent the server’s cache from growing, use recursion no; in combination with fetch-glue no;.</td></tr>
</table>

Table 1-4 NAMED.CONF Options (Continued)

Option	Description
topology { address_match_list; }	Specifies the servers network topology in order to determine what server to query when recursively following a referral. Each element is given a rating based on the position in the list, based on descending preference. Addresses that match up with listed networks are queried first, followed by addresses that match up with unlisted networks, followed by addresses that match up with negated networks. Example: `topology { !10.10/16; 10/8; 127.0.0/24; };` Addresses in 10.*.*.* will be preferred most of all, with the exception of 10.10.*.* addresses. Addresses in the 127.0.0.* will be preferred next. All other addresses will be preferred over any 10.10.*.* addresses.

Address_match_list

The following can be address match lists:

- an IP address (in dotted-decimal notation)
- an IP address match list
- an IP prefix (in /- notation)
- an address match list defined with the *acl* statement

The following ACLs are predefined:

- any
- none
- localhost
- localnets

Place the ! character in front of elements you want to negate.

When an IP address or prefix is compared to an address match list, the list is examined and the first match (regardless of its negated state) is used. The interpretation of a match depends on the conditions defined in the following table.

When a list is being used...	A non-negated match...	A negated match...
as an access control list	allows access.	denies access. You can use the *listen-on* clause to define local addresses not normally used to accept nameserver connections.
with the topology clause	returns a distance based on its position on the list; the closer the match to the start of the list, the shorter the distance between it and the server.	A negated match is assigned the maximum distance from the server. ***Note!*** 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.

Since the address match list uses a first-match algorithm, care must be taken when using negation. In general, if an element is a subset of another element, the subset should be present in the list before the broader element.

For example, 10.0.0/24; !10.0.0.1 will never negate to the 10.0.0.1 address because a 10.0.0.1 address will match with the 10.0.0/24 element and not traverse any farther. So the 10.0.0.1 address will be accepted in the match list.

Using !10.0.0.1; 10.0.0/24 will elicit the desired effect. The 10.0.0.1 will be matched against the first, negated, element. All other 10.0.0.* addresses will pass by the 10.0.0.1 element and be matched against the 10.0.0/24 subnet element.

Logging

The *logging* section configures a wide variety of logging options for the nameserver. Its channel phrase associates output methods, format options and severity levels with a name that can be used with the category phrase to select how various classes of messages are logged. The basic *logging* syntax is as follows:

```
logging {
        channel <channel_name> {
            file <pathname>;
            severity <type>;
            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 section is used to define as many channels and categories as you want. If there are multiple *logging* sections in a configuration, the first one defined determines the logging, and warnings are issued for the others. If there is no logging section, the default logging configuration will be:

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

The following is an example:

```
channel xfers {
     file “TCPWARE:XFERS.LOG”;
     severity info;
     print-severity yes;
     print-time yes;
};
     category xfer-in {
     xfers;
};
```

Table 1-5 describes the logging options.

Table 1-5 Logging Options

Options	**Description**
channel	Specifies where the logging data goes: to syslog, to a file, to stderr, or to null.
category	Specifies what data is logged. You can send a category to one channel or to many channels. These are the valid categories: default update notify panic packet config ncache cname load maintenance parser xfer-in security eventlib statistics queries xfer-out os insist db lame-servers
file	Specifies the path name of the file you want the message to go to.
syslog daemon	Specifies that the message goes to syslog (opcom) instead of to a file.
severity	Specifies the severity level for this channel. The severity choices are critical, error, warning, notice, info, debug [level], and dynamic.

Table 1-5 Logging Options (Continued)

Options	Description
print-category print-severity print-time	Specifies the category, severity level, and time stamp of the messages. The default is NO for each item.

Caching Data in Memory

The TCPware_NAMED process caches a certain amount of DNS data in memory. By default, it limits the maximum time-to-live (TTL) on a resource record placed in memory cache to one week. If the TTL is set to more than a week, it only caches it for a week. This value may be too high for some systems with heavily used name servers and limited memory.

When NAMED starts up, it checks the SYSTEM EXECUTIVE logical table for the TCPWARE_NAMED_MAX_CACHE_TTL logical value and sets the maximum cache time (in seconds) to be that value. You can use this logical to override the default one week (604800 seconds) to a maximum cache time more appropriate for your system. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_NAMED_MAX_CACHE_TTL 86400
```

This logical is read the next time the server starts. If you do not want to wait for the server to start, you can make the change to the running server by using the NETCU SET NAMED MAX_TTL <ttl> command. See Chapter 2 of the *Network Control Utility (NETCU) Command Reference*. Any data now written to the cache remains there for 86400 seconds (one day).

Editing Database Files

The CNFNET configuration procedure creates templates of the database files. Some of the templates are only examples of the database records necessary for configuration. You must enter information specific to your configuration for the server to function properly. The type of server you configure determines which database file you need to edit.

Note! If you edit the database files, stop and restart the Domain Name Services software so that the name server can update its database. See *Starting and Stopping*.

Special Characters

The characters listed in Table 1-6 have special uses in the database files.

Table 1-6 Special Characters

The character...	**Is used...**
char	To quote a single character, *char*, that otherwise has special meaning. For example, use to place a dollar sign in the text.
( )	To group data that exceeds a line boundary. The SOA record requires parentheses. See the Start of Authority (SOA) resource record.
;	To start a comment. TCPware ignores the remaining characters in the line.
*	As a wildcard. For example, the domain name *.FLOWER.COM refers to any host within the FLOWER.COM domain.

Use a period (.) to end a domain name. If the domain name does not end with a period, Domain Name Services appends the current domain name to it. The NAMED.CONF file defines the current domain name, or you can use the `$origin` command to redefine it.

Special Commands

You can use two special commands in database files:

- **`$origin`**
- **`$include`**

These commands are described on the following pages.

$origin

Indicates that all the records following the command belong to a different domain than the previous records. The name server has authority over all records listed. This command gives you a shorthand way of expressing the domain name of the host.

Format

$origin *domain-name*

Parameter

domain-name

Domain name to which the records belong.

Example

This example defines the DAISY.FLOWER.COM, IRIS.FLOWER.COM, SPARROW.BIRD.COM, DOVE.BIRD.COM, MAPLE.TREE.COM, and ACORN.TREE.COM domains.

```
;the default domain is flower.com.
daisy       in    a    192.168.62.1
iris        in    a    192.168.62.2
$origin bird.com.
sparrow     in    a    192.168.95.1
dove        in    a    192.168.95.3
$origin tree.com.
maple       in    a    192.168.74.1
acorn       in    a    192.168.74.2
```

$include

Includes an external file in a DNS database file. Useful for organizing different types of information into separate files.

Format

$include ***file [origin]***

Parameter

file

File you want to include in the database.

origin

(Optional) Origin of the include file (see the `$origin` command on the previous page).

Examples

1 This command includes the file TCPWARE:MAILLIST.TXT in the database file.

```
$ include tcpware_named_root:maillist.txt
```

2 These commands include the file TCPWARE:BIRD.TXT in a series of definitions, using the BIRD.COM domain as origin (see the `$origin` command) for the include file.

```
$ origin flower.com.
daisy
iris
$ include tcpware_named_root:bird.txt bird.com.
rose
```

If the BIRD.TXT file includes records for SPARROW and DOVE, the extended version of the above definition would be:

```
daisy.flower.com
iris.flower.com
sparrow.bird.com
dove.bird.com
rose.flower.com
```

Note that the include file origin definition is for the include file only and does not affect the "external" origin.

Resource Records

All database files contain entries called resource records. TCPware for OpenVMS supports the resource record types listed in Table 1-5 under the *type* record field.

This section provides the following information for each DNS resource record:

- Purpose
- Format
- Field definitions
- Example of usage

The fields that are common to most resource records are: *owner*, *ttl*, *class*, and *type*. The remaining *data* fields are different for each record type. This section documents these *data* fields under the appropriate record type.

All resource records are case insensitive. However, TCPware preserves the case you enter.

Format

owner ttl class type data

Fields

owner

Domain name of the owner of the record.

The domain name can be absolute or relative. An absolute domain name lists all the labels of the name and ends with a period. For example, `DAISY.FLOWER.COM.` is an absolute domain name. A relative domain name does not end with a period. DNS assumes it belongs to the current domain. For example, `DAISY` is a relative domain name in the FLOWER.COM domain.

Acceptable characters are A through Z (upper or lower case), 0 through 9, and dash (-). The period (.) is a label separator.

The values listed in Table 1-7 have special meaning in the *name* field.

Table 1-7 Special Values for *name*

Value	Description
(all blank)	The resource record applies to the last explicitly stated domain name
.	Indicates the root (or top level) domain
@	Indicates this is the current domain name

ttl

Time-to-live (TTL).

This is the length of time (in seconds) the record is valid after a requestor host receives it from a primary server. For example, a TTL of 86400 equals 24 hours.

You can specify in the time-to-live field in the following ways:

Note! Each of these is equivalent to one week.

- 604800
- 1w
- 7d
- 168h
- 10080m
- or any combination

For example: `sigma 2h46m40s IN A 192.1.1.97`

Loads the TTL as: `ttl = 10000 (2 hours 46 mins 40 secs)`

If you leave this field blank, DNS uses the TTL designated in the SOA (start of authority) record *minimum* field. The *minimum* field value is the "default" TTL.

All resource records that have the same values in the *name*, *class,* and *type* fields should also have the same value in the *ttl* field.

class

Address class (should be **`IN`** for "internet").

type

Resource record type, as listed in Table 1-8 and described more fully on the following pages.

Table 1-8 Resource Record Types

Type	**Description**	**Type**	**Description**
A	Address	MX	Mail exchanger
ASFDB	AFS database	NS	Name server
CNAME	Canonical name	PTR	Pointer
HINFO	Host information	RP	Responsible person
ISDN	ISDN information	RT	Route through binding
MB	Mailbox	SOA	Start of authority
MG	Mail group	TXT	Descriptive text
MINFO	Mail information	WKS	Well known services

Table 1-8   Resource Record Types

Type	Description	Type	Description
MR	Mailbox rename	X25	X.25 information

data

Data specific to each entry.

This field varies with each type of resource record.

A

Address record, or internet address of a host. The name server uses this record when it responds to a query for an internet address.

Use this record in any of the database files.

Format

owner ttl class A address

Fields

A

The *type*, which must be **A**.

address

Internet address of the host specified in the *name* field.

Example

This example includes two A records for local hosts DAISY and LILAC.

```
;name      ttl        class      type     address
daisy                 IN         A        192.168.95.3
lilac      99999999   IN         A        192.168.95.4
```

AFSDB

AFS Database (AFSDB) record. The AFSDB record was added as of RFC 1183 for experimental purposes. AFS (which stands for the Andrew File System designed at Carnegie-Mellon University in Pennsylvania) is a network file system like the Network File System (NFS). The AFS database (AFSDB) is an authenticated name server for AFS filesystems.

The AFSDB record finds the location of the AFS name server.

Use this record in any of the database files.

Format

owner ttl class **AFSDB** ***subtype*** ***host***

Fields

AFSDB

The *type*, which must be **`AFSDB`**.

subtype

Either **`1`**, for an AFS cell database server, or **`2`**, for a DCE-authenticated name server.

host

Domain name that specifies a host that has a server for the cell named by the resource record's owner.

Example

This example is an AFSDB record for host DAISY.

```
;owner    ttl    class    type    subtype    host
daisy            IN       AFSDB   1          daisy.flower.com.
```

CNAME

Canonical Name (CNAME) record, or official name of the host. You can include a nickname, or if you rename the host, use the nickname field to give the old domain name.

Note! Do not use the nickname in any other resource records.

Format

nickname ttl class **CNAME** ***canonical-name***

Fields

nickname

Nickname or alias for the host. If you rename the host, this is the old domain name.

CNAME

The *type*, which must be **`CNAME`**.

canonical-name

Official domain name for the host. This can be an absolute or relative name. If you rename the host, this is the new domain name.

Example

This example includes two CNAME records. These records define SPRING as a nickname for host LILAC.FLOWER.COM and SUMMER as a nickname for host DAISY.FLOWER.COM. Because no period (.) follows the nicknames, DNS assumes they are in the current domain.

```
;nickname    ttl    class     type    canonical-name
spring              IN        CNAME   lilac.flower.com.
summer              IN        CNAME   daisy.flower.com.
```

HINFO

Host Information (HINFO) record, or hardware type and operating system of a host. DNS uses this information to answer queries.

Each host in a domain can have just one HINFO record.

Format

owner ttl class **HINFO** ***hardware opsys***

The *hardware* and *opsys* fields require a space between them. If you need to use a space within either field, enclose the field within quotation marks (`" "`).

Fields

HINFO

The *type*, which must be **`HINFO`**.

hardware

Type of CPU. See the latest *Assigned Numbers* RFC for a list of standard hardware types.

opsys

Host operating system.

Example

This example includes two records for hosts IRIS and LILAC in the current domain, and gives their hardware type and operating system. Some field entries include quotation marks because they contains space characters.

```
;owner    ttl    class    type   hardware              opsys
iris             IN       HINFO  "AlphaStation 8400"   VMS
lilac            IN       HINFO  "VAXStation 4000"     "VMS V7.0"
```

ISDN

Integrated Service Compaq Network (ISDN) record, or telephone number of a host.

Format

owner ttl class **ISDN** ***ISDN-address [sa]***

Fields

ISDN

The *type*, which must be **`ISDN`**.

ISDN-address

Decimal ISDN number and direct dial-in (DDI), if necessary, of the *owner* field.

sa

Optional sub-address in hex digits.

Example

This example gives the ISDN addresses for the specified hosts.

```
;owner           ttl     class     type    ISDN-address      sa
Relay.Prime.COM.         IN        ISDN    150862028003217
sh.Prime.COM.            IN        ISDN    150862028003217   04
```

MB

Mailbox (MB) record, or Internet mailbox that is the subdomain of a domain. This record is usually no longer used.

Format

owner ttl class **MB** ***mailbox***

Fields

MB

The *type*, which must be **`MB`**.

mailbox

Subdomain name of the host with the specified mailbox.

Example

This example gives the mailbox record for host TULIP.

```
;owner    ttl    class     type    mailbox
tulip            IN        MB      mailbox.tulip.flower.com.
```

MG

Mailbox Group (MG) record, or Internet mailbox group (mailing list) that is the subdomain of a domain. This record is usually no longer used.

Format

owner ttl class **MG** ***mail-group***

Fields

MG

The *type*, which must be **`MG`**.

mail-group

Subdomain name of the mailbox belonging to the mail group specified by the domain name.

Example

This example gives the mailbox group record for host TULIP.

```
;owner    ttl     class    type    mail-group
tulip             IN       MG      mailbox.tulip.flower.com.
```

MINFO

Mail Information (MINFO) record, or Internet mailbox that is responsible for the mailbox or mailing list or that receives error messages related to the mailbox or mailing list.

Format

owner ttl class **MINFO** ***resp-mbox*** ***error-mbox***

Fields

MINFO

The *type*, which must be **`MINFO`**.

resp-mbox

Domain name mailbox responsible for the mailbox or mailing list.

error-mbox

Domain name mailbox that receives error messages for the mailbox or mailing list specified by the owner of the MINFO record.

Example

This example gives the mail information record for host TULIP.

```
;owner ttl class  type   responsible             errors
tulip      IN     MINFO  mailer.tulip.flower.com.err.tulip.flower.com.
```

MR

Mailbox Rename (MR) record that renames an internet mailbox that is the subdomain of a domain. This record is usually no longer used.

Format

owner ttl class **MR** ***new-mailbox***

Fields

MR

The *type*, which must be **`MR`**.

new-mailbox

New name of the domain name of the host with the specified mailbox.

Example

This example gives the mailbox rename record for host TULIP.

```
;owner    ttl    class    type    mailbox
tulip            IN       MR      newbox.tulip.flower.com.
```

MX

Mail Exchanger (MX) record, or a host that can accept mail for another host. A host can have multiple MX records. Each record receives a preference value.

When a mailer tries to deliver mail to a host:

1	It reads all MX records defined for the destination host and sorts them by preference value.
2	It tries to deliver the mail to the host specified on the record with the highest preference. The record with the lowest value (beginning at 0) has the highest preference.
3	If the first attempt fails, it tries the host specified on the record with the next highest preference value.
4	It keeps trying until it delivers the mail, or until it tries the host specified on the record with the lowest preference.

If you assign the same preference value to multiple MX records for a host, the mailer tries the equally-preferenced records in random order.

See Chapter , *Managing Mail Services*, the *Mail Exchanger Records* section for MX records used with the SMTP mail facility.

Format

system *ttl* *class* ***MX*** ***preference*** ***gateway***

Fields

system

Domain name of the host the *gateway* host accepts. The host might not be connected directly to the network. Using wildcards in domain names is strongly discouraged if the hosts are Internet-connected (see section 2.7 of RFC 1912 for details).

MX

The *type*, which must be **`MX`**.

preference

Delivery order when a host has multiple MX records. The lower the number (starting at 0), the higher the preference.

gateway

Name of the host accepting mail for the host specified in the *system* field.

Example

This example gives three MX records for host TULIP (in the current domain). The mailer tries to deliver the mail to host TULIP.FLOWER.COM first, because that record has the lowest *preference*-value. If the attempt fails, it tries host IRIS.FLOWER.COM and then LILAC.FLOWER.COM.

```
;system    ttl    class    type    preference    gateway
tulip             IN       MX      10            tulip.flower.com.
tulip             IN       MX      15            iris.flower.com.
tulip             IN       MX      20            lilac.flower.com.
```

NS

Name Server (NS) record that lists the domain name of a host that provides domain name services, and the name of the domain being served. Therefore, the specified host is an authoritative name server for the specified domain.

You can enter NS records in any database file.

Format

owner ttl class **NS** ***server***

Fields

NS

The *type*, which must be **`NS`**.

server

Domain name of the host that serves the domain.

Example

This example gives the syntax of three NS records. DAISY.FLOWER.COM and IRIS.FLOWER.COM are both servers for the FLOWER.COM domain. The *owner* field is blank for IRIS.FLOWER.COM, indicating it serves the domain specified in the previous record. The *owner* field for NS.NASA.GOV server contains only a dot (.), indicating NS.NASA.GOV is a root server. IRIS.FLOWER.COM takes its time-to-live (TTL) value from the *min* field of the SOA record. The TTL for NS.NASA.GOV is 99999999 seconds (approximately three years).

```
;owner     ttl      class      type      server
flower.com.         IN         NS        daisy.flower.com.
                    IN         NS        iris.flower.com.
.          99999999 IN         NS        ns.nasa.gov.
```

PTR

Domain Name Pointer (PTR) record that allows special names to point to another location in the domain. The most common use of PTR records is for reverse mapping: Domain Name Services finds a host domain name when given an internet address. The IN-ADDR.ARPA domain maintains reverse mapping information.

PTR records in the IN-ADDR.ARPA domain contain a host name that consists of the internet address specified in reverse order, combined with the IN-ADDR.ARPA domain name. This name points to the domain name of the host with that internet address.

Enter PTR records in the NAMED.REV, NAMED.HOSTS, or NAMED.LOCAL files.

Format

rev-addr *ttl* *class* **PTR** ***realname***

Fields

rev-addr

Combination reverse internet address and domain name IN-ADDR.ARPA. Each *rev-addr* should be unique to the zone.

PTR

The *type*, which must be **`PTR`**.

realname

Full domain name of the host. If the host is not in the current domain, this name must end in a period (.). Do not use a nickname.

Example

This example gives PTR records for two hosts. The internet addresses are in the current domain, 95.168.192.IN-ADDR.ARPA.

```
;rev-addr   ttl    class    type    realname
$origin 95.168.192.in-addr.arpa
2                  IN       PTR     daisy.flower.com.
4                  IN       PTR     lilac.flower.com.
```

RP

Responsible Person (RP) record for the name of the person responsible for a host or domain. You can specify multiple responsible persons for a domain name, but make sure that the *ttl* for the records are the same.

Format

owner ttl class **RP** ***mailbox*** ***txt-domain***

Fields

RP

The *type*, which must be **RP**.

mailbox

Domain name of the mailbox for the responsible person.

txt-domain

Domain name where TXT records exist.

Example

This example gives RP records for two hosts.

```
;owner   ttl   class  type  mailbox             txt-domain
flower   6666  IN     RP    honcho.flower.com. daisy.flower.com.
         6666         RP    kahuna.flower.com. iris.flower.com.
```

RT

Route Through (RT) record that provides a route-through binding for hosts that do not have a direct WAN address. Use this record in much the same way as the MX resource record.

Format

owner ttl class **RT** ***preference*** ***interm-host***

Fields

RT

The *type*, which must be **RT**.

preference

Preference of the route, in a 16-bit integer. The smaller the number, the more preferred the route.

interm-host

Name of the intermediate host domain that serves as an intermediary in reaching the host specified by *owner*. The DNS resource records associated with *interm-host* must include at least one A, X25, or ISDN record.

Example

This example shows the route-through for two hosts. When a host looks up DNS records to try to route a datagram, it first looks for RT records for the destination host, which point to hosts with address records (A, X25, or ISDN) compatible with the WAN available to the host. If the host itself is in the set of RT records, it discards any RTs with preferences higher or equal to its own. If there are no remaining RTs, it can use address records of the destination itself.

```
;owner   ttl   class   type   preference   interm-host
iris           IN      RT     2            marigold.flower.com.
daisy          IN      RT     10           pansy.flower.com.
```

SOA

Start of Authority (SOA) record that defines the start of a zone. There is one SOA record for each zone and it is on the primary server. If other servers in the zone have SOA records, these records must be identical to the one on the primary server. The SOA record is the first one listed in a database file.

You can enter this record in any database file. The NAMED.CA file can store an SOA record, but the record does not define the server as authoritative.

Format

owner ttl class **SOA** ***origin*** ***person*** **(serial refresh retry expire minimum)**

The parentheses are required if continuing onto one or more subsequent lines. At least one space must separate the parentheses from the text within it.

Fields

SOA

The *type*, which must be `SOA`.

origin

Name of the host on which the primary server resides.

If the local host is not the primary server, the local host periodically obtains database information from the specified host. See the *refresh, retry*, and *expire* fields.

person

Mailbox address of the person responsible for the DNS software on the local domain. Replace the `@` sign in the mailbox address with a period (**.**); for example: `gardener@iris.flower.com` becomes `gardener.iris.flower.com.`.

serial

Version number of the database file. A 32-bit unsigned integer that can theoretically start at 0.

Increment this field by a certain interval each time you edit a database file (using the YYYYMMDDVV date syntax provides a "safe" interval). If *serial* on the primary server is "higher" (based on serial number arithmetic) than *serial* on the secondary server, the secondary server knows that the primary server contains new data and it performs a zone transfer to update its database. The serial number also tells the DNS software which of two file copies is the most recent.

refresh

Time interval (in seconds) after which the secondary server must request the SOA record from the primary server. A 32-bit unsigned integer, measured in seconds. For example, a refresh value of 86400 equals 24 hours. A value of 900 seconds (15 minutes) is the minimum value allowed.

retry

Time interval (in seconds) after which the secondary server should re-request the SOA record from the primary server after a refresh failure. A 32-bit unsigned integer, measured in seconds. 600 seconds (10 minutes) is a reasonable value.

expire

How long the secondary server can use its copy of the database file when it cannot obtain a refresh. A 32-bit unsigned integer, measured in seconds. A typical value is 3600000 seconds (approximately 41 days and 16 hours).

minimum

Minimum time to live (TTL) value, in seconds, for the records in the zone. DNS uses this value if you do not specify the *ttl* field for other resource records. A reasonable value is 86400 seconds (24 hours).

Example

This example shows a typical SOA record format. The values are described in the table.

```
;owner  ttl  class  type  origin            person
@            IN     SOA   iris.flower.com. gardener.iris.flower.com.(
                    1            ; serial
                 3600            ; refresh (1 hour)
                  600            ; retry (10 minutes)
              3600000            ; expire (1000 hours)
              86400 )            ; minimum (24 hours)
```

Value	Description
@	Current domain name
iris.flower.com.	Primary server host name
gardener.iris.flower.com.	"Owner" of the DNS software on the local domain (the mailbox address gardener@iris.flower.com becomes gardener.iris.flower.com.)
1	Serial (version) number of the database file
3600	Refresh time – the secondary server requests an SOA record from the primary server every 3600 seconds (1 hour)
600	Retry time – the secondary server retries requests for the SOA record from the primary server every 600 seconds (10 minutes) if a refresh fails

Value	**Description**
3600000	Expiration time – the secondary server can use its copy of the database if all refreshes fail for a total of 360000 seconds (1,000 hours, or 41 days and 16 hours)
86400	Minimum time-to-live of 86400 seconds (24 hours) for records in the zone

The parentheses at the end of the second line indicate that one or more additional lines related to the record follow. The lines usually include the *serial*, *refresh*, *retry*, *expire*, and *minimum* field values and their commented out (;) descriptions. You must include a space between the parentheses and the text it encloses (such as by indenting the next line). You must also include a white between the last value (86400 in the example) and the closing parentheses.

TXT

Text (TXT) record. Holds descriptive text. The semantics of the text depend on the domain.

Format

owner ttl class **TXT** ***txt-data***

Fields

TXT

The *type*, which must be **TXT**.

txt-data

One or more character strings of descriptive text.

WKS

Well Known Service (WKS) record. Lists the well known services a specified host provides on port numbers below 256. DNS uses this information to answer queries.

Enter WKS records in the NAMED.HOSTS file.

Format

owner ttl class **WKS** ***address protocol services***

Fields

WKS

The *type*, which must be **`WKS`**.

address

IP address of the host. Define one WKS record for each protocol at each internet address.

protocol

Name of the generic protocol that supports the well known services. Must be **`UDP`** or **`TCP`**.

services

Well known services the protocol provides, such as FTP, TELNET, SMTP, or SUNRPC.

Example

This example gives protocol information for hosts IRIS and DAISY in the current domain.

```
;owner  ttl  class   type   address           protocol services
iris         IN      WKS    192.168.95.1 tcp   ftp telnet smtp
daisy        IN      WKS    192.168.95.3 udp   unrpc
```

X25

X.25 address of a host.

Format

owner ttl class **X25** ***PSDN-address***

Fields

X25

The *type*, which must be **`X25`**.

PSDN-address

Public Switched Data Network (PSDN) address (in decimal digits) in the X.121 numbering plan associated with the *owner* field. The string of digits begins with a four-digit Data Network Identification Code (DNIC). Do not use national prefixes in the address.

Example

This example gives the PSDN address of the specified host.

```
;owner             ttl  class  type  PSDN-address
lily.flower.com.        IN     X25   311061700956
```

Troubleshooting Domain Name Services

Access error messages help by entering **`HELP TCPWARE MESSAGES`** ***`[identifier]`***, or connect to web site **`http://www.process.com`** (select **`Customer Support`** followed by the **`Error Messages`** button).

When you start the name server, it creates a log file, TCPWARE:NAMESERVER.LOG. This file contains information about any problems uncovered when loading the database or during name server operation. TCPware sends system logging errors to this file. By default, TCPware logs all errors except Debug messages to the log file.

Dynamic TCP/IP Load Balancing

TCPware provides TCP/IP load balancing services for a TCP/IP cluster that are analogous to the load balancing services the LAT terminal service provides.

When a new TCP connection to a cluster name occurs, the TCPware Domain Name Services name server assigns the connection to one of a number of hosts. The host to which it assign the connection depends on:

- The availability of the host.
- The observed load on the host.

TELNET most often uses TCP/IP cluster load balancing, although other TCP protocols do also. UDP-based protocols also work well with cluster load balancing, but only if:

- The server (such as DNS) does not retain state information.
- The client (such as TFTP) resolves the domain name only once at the start of a connection.

TCP/IP load balancing does NOT:

- Provide effective NFS server failover. Most clients do not resolve names again and remount filesystems when an NFS server fails to respond.
- Provide local preference to clients' selection of hosts.
- Actively re-balance the load: a failed and recovered host receives only new connections.
- Support non-TCPware hosts as part of the cluster.

In addition, the default metric is not very useful for RPC type services. It is oriented toward measurement of users.

Configuration Requirements

All hosts in the TCP/IP cluster must run TCPware. Also, all DNS servers for the zone defined with the TCP/IP cluster name must be on systems running TCPware. Client systems do not have to run a particular TCP/IP implementation.

The term *cluster* here means a *TCP/IP cluster*. Hosts in a TCP/IP cluster do not have to be part of a VMScluster. They even do not have to be on the same bridged LAN.

In a TCP/IP cluster, the hosts can be at least one of the following:

- Independent systems with TCP/IP connectivity

- Located anywhere so long as there is TCP/IP connectivity
- Part of several VMSclusters with TCP/IP connectivity

You can define multiple cluster names describing subsets, or overlapping or separate clusters.

Load Balancing Process

When a client wants to connect to a host within the cluster:

1	It sends the DNS server a name-to-address translation request for a host to which it wants to connect.
2	The DNS server looks in a cache that holds recent load information for the hosts in the cluster. If the name is a cluster name, a routine sorts the addresses by reported load.The server determines the load by exchanging UDP datagrams with each host in the cluster, which report their current load metrics. The server treats hosts that fail to respond as unavailable and does not offer them any new traffic. Specifically: **a** The DNS server searches its resource record for host addresses and looks up each host in a private list of hosts. Different cluster names share this list. If a host appears in more than one cluster, the server requests its load metrics only once. **b** The DNS server sends update requests to the hosts if there is no information or if there is outdated information in the cache. **c** If a host fails to respond to the load information requests, it does not return its address to the server. In this case, the host could actually be down. Alternatively, the host also could have been administratively shut down by removing the UDP service that responds to load requests. Removing this UDP service effectively removes the host from the cluster. **d** TCPware moves the address record corresponding to the host with the lowest load metric to the front of the DNS information list.
3	The server responds to the client with a list of addresses in preferred order of use. Most clients use the first address, or if it fails, second or subsequent addresses.
4	When DNS returns its reply to the client, it: • Rotates the first address down the list among hosts with similar load metrics. This means that DNS "round-robins" calls among similarly loaded hosts. • Sets the time to live (TTL) to a fairly small value. This forces a new request for subsequent connections.

Cluster Names

In DNS, you define each cluster name as an ordinary host name with an IP address and resource records for each host address used. There are straightforward, overlapping, and subzone clusters.

Straightforward Clusters

Example 1-1 shows a cluster name defined as a DNS address (A) resource record as part of a zone file. The cluster is called `perennials` and is assigned IP address 192.168.3.50.

Example 1-1 Cluster Name in the Zone File

```
@              IN      SOA     rose.nene.com. system.rose.nene.com. (
                                  1        ; Serial
                                  3600     ; Refresh
                                  600      ; Retry
                                  3600000  ; Expire
                                  86400 )  ; Minimum
               IN      NS      rose.nene.com.
rose           IN      A       192.168.3.50
lilac          IN      A       192.168.3.51
petunia        IN      A       192.168.3.52
hydrangea      IN      A       192.168.3.53
perennials     IN      A       192.168.3.50
               IN      A       192.168.3.51
               IN      A       192.168.3.52
               IN      A       192.168.3.53
```

As the DNS server consults its cache, it matches the cluster name against a list. If the name is in the list, TCPware sorts the addresses in the list by reported load.

Overlapping Clusters

Example 1-2 shows two clusters defined in a zone file. Note that the `192.168.100.2` address is common to both clusters.

Example 1-2 Overlapping Clusters in the Zone File

```
$ORIGIN nene.com
orders    IN  A   192.168.100.1
          IN  A   192.168.100.2
invoices  IN  A   192.168.100.2
          IN  A   192.168.100.4
```

Subzone Clusters

In some cases, you may want to load balance your cluster using an external name server instead of a local one. Since the external server cannot actually configure an internal load balanced cluster, the primary server must delegate authority on a subdomain to the internal server. The internal server then becomes primary for the subzone, which becomes the actual address of the cluster.

The following set of examples show three systems in a cluster: 10.0.0.1, 10.0.0.2 and 10.0.0.3. The domain is `flower.com`. The first step is to set up a subdomain on the primary server by editing the zone file for the `flower.com` domain and adding the line to delegate the authority to the internal server, `homerdns.flower.com` (see Example 1-3).

Example 1-3 Delegating Authority to a Subzone for a Cluster

```
homer    IN  NS  homerdns.flower.com.
```

The next step is to set up `homerdns.flower.com` as a primary name server for domain `homer.flower.com`.

The zonefile for `homer.flower.com` on `homerdns.flower.com` should include the lines in example 1-4.

Example 1-4 NAMED.CLUSTER File

```
cluster  IN  A   10.0.0.1
         IN  A   10.0.0.2
         IN  A   10.0.0.3
```

There is now a load balanced cluster set up to be `cluster.homer.flower.com` that is accessible from both the primary server (by delegation) and the internal server.

Finally, map `cluster.flower.com` to the load balanced cluster. Add the following line to the NAMED.HOSTS equivalent file on the primary server:

```
cluster  IN  CNAME  cluster.homer.flower.com.
```

The primary server now serves out the addresses in Example 1-4 in load balanced order.

Load Request Protocol

When the DNS server finds that its load information for a host is out of date, it sends a UDP datagram to the host asking for load updates.

Each UDP request starts a timed sequence in the host. This causes the host to send updates to the DNS server at specified intervals over a set period of time. When the sequence ends, the DNS server considers the information stale and sends a new request. This procedure:

- Minimizes traffic when the DNS server is heavily loaded (for example, handling more than 100 requests per second from clients).
- Is quiet when there are no requests.

The procedure does not require hosts to maintain more than a transient state on the DNS servers, since if they fail, they simply cease to respond.

If a host is part of multiple clusters, the DNS server makes the load request once and not for each separate cluster.

The host normally provides the load reply service from within NETCP. However, you can do this through a configured UDP service using the definition:

```
NETCU> ADD SERVICE METRIC UDP /ROUTINE=REPORT_TCLB_METRIC
```

LAT looks at the number of processes in COM state and uses that information to calculate its metric. LAT determines the TCP/IP cluster load balancing metric from the number of active users on the system.

Part of the metric consists of a value that is set for each host. You set this value by defining the system logical TCPWARE_TCLB_BIAS with a multiplier and an addend as two values of the logical. Both are real numbers. TCPware uses the values in computing the reported load.

You can also use these values to bias a load offered to the host. For example, the following command doubles the observed load and adds 1.5 users:

```
$ DEFINE/SYSTEM TCPWARE_TCLB_BIAS "2.0","1.5"
```

A cluster might consist of four hosts with one running other tasks. This host should not receive its full share. You can set the values to cause that host to report a higher load.

TCPware re-translates the TCPWARE_TCLB_BIAS logical before it sends each response. This means that some other process can change it dynamically or you can set it statically.

Chapter 2

DHCP/BOOTP Server

Introduction

This chapter describes the DHCP/BOOTP Server. It combines the Dynamic Host Configuration Protocol (DHCP) server with the bootstrap protocol daemon (BOOTPD). It is divided into the following sections:

- DHCP and BOOTP
- DHCP Process
- DHCP Administration
- DHCP Configuration
- DHCP Conversion Tool
- Introducing the DHCP Configuration File
- Verifying the DHCP Configuration
- Reloading the DHCP Configuration
- DNS Dynamic Updates Within DHCP
- Host Name Generation
- Configuration File Declarations and Parameters
- DHCP Options
- DHCP Lease Format
- Working with DHCP Leases
- Address Lease States in DHCP Dump Files
- DHCP Safe-failover Introduction
- Configuring Safe-failover
- Boot File for Safe-failover
- State File for DHCP Safe-failover
- DHCP Safe-failover Lease File Statements
- DHCP Safe-failover Configuration File Statements
- Transitioning to DHCP Safe-failover Partner Down State

DHCP and BOOTP

DHCP is the Dynamic Host Configuration Protocol. It centralizes and automates TCP/IP network configuration. The DHCP Server dynamically allocates IP addresses for hosts on the network from an available pool of addresses. In this way, new hosts or hosts that are frequently relocated can automatically get new IP addresses for a certain lease period.

DHCP is an extension of the Internet Bootstrap Protocol (BOOTP). DHCP offers a network host a temporary lease rather than an ownership of an IP address. The lease identifies the duration for which the client can safely use its dynamically assigned IP address. Lease lengths generally depend on the number of network users (crowding of the network) and the number of available IP addresses the DHCP server can assign. The network manager sets the lease length through parameters.

BOOTP uses UDP/IP to allow diskless systems to find their IP addresses, addresses of boot servers, and names of boot files. BOOTP can supply other client information, such as the addresses of name servers, gateways, and LPD servers.

In TCPware, the DHCP Server (DHCPD) is combined with the BOOTP Server (BOOTPD) to form the DHCP/BOOTP Server (DHCPD/BOOTPD).

Note! DHCP uses DNS for host names and IP addresses; thus, a malfunction in your DNS server can affect the DHCP server.

DHCP Process

DHCP goes through an initializing, selecting, requesting, binding, renewal, rebinding, and expiration cycle when negotiating for an IP address, as shown in Figure 2-1. The process is basically as follows:

1	The client just added or relocated on the network requests an IP address by broadcasting a DHCPDISCOVER message to the local subnet over the well-known BOOTP server port. (The client can also go through a BOOTP router or relay agent to forward the DHCPDISCOVER to additional remote DHCP servers.) This is the initializing state.
2	The participating DHCP servers respond with a DHCPOFFER message if they have a valid configuration for the client. The client may get many of these messages, which contain the IP address and configuration data. (The servers make sure to reserve the addresses so as not to accidentally offer them to another client.) At this point the client enters the selecting state.
3	After selecting an address, the client broadcasts the selected address and name of the "winning" server (`DHCP Server 1` in Figure 2-1) using a DHCPREQUEST message. This is the requesting state. All the other servers can now safely unreserve their addresses.

4	Server 1 sends the client a DHCPACK (acknowledgment) message with the negotiated IP address, the lease, and the network configuration parameters. The client now enters the binding state and can fully use the assigned IP address.
5	About halfway through the lease, the client sends Server 1 another DHCPREQUEST for a lease renewal, and enters the renewal state. If the server deems the lease renewable, it sends back another DHCPACK to update the lease (including any new parameters). The client now returns to the binding state, as in Step 4.
6	If the client cannot renew the lease (such as if Server 1 is down), the client waits until about 87.5% of the way through the lease and broadcasts another DHCPREQUEST to all DHCP servers. Any server can now return a DHCPACK containing the extended lease and updated parameters. This is the rebinding state.
7	When the lease reaches 100% expired, or a server sends back a DHCPNAK negative acknowledgment message, the client must give up the IP address. It then returns to the initializing state and has to start the address negotiation over again.

Figure 2-1 DHCP Address Request and Allocation Process

DHCP is defined in RFC 2131 and RFC 2132. Refer to them for more information.

Two DHCP servers are recommended for a network. The benefit of having more than one server is if one fails another is available to continue processing requests, ensuring that all hosts (old and new) are serviced continuously. Refer to *DHCP Safe-failover Introduction* for more information.

DHCP Administration

You can administer the DHCP server using the following TCPware Network Control Utility (NETCU) commands:

Command	**Description**
RELEASE DHCP *ip-address*	Forces the DHCP server to act as if it heard a DHCPRELEASE message from the client for the given IP address.
REMOVE DHCP *ip-address*	Synonym for RELEASE DHCP.
SET DHCP/DEBUG=*value*	Sets the debug logging level to the given value.
SET DHCP/NEWLOG	Starts a new debug log file.
SET DHCP/PARTNERDOWN	For Safe-failover DHCP: causes the DHCP server to transition into Partner Down state.
STOP/DHCP	Causes the server to shut down.
SHOW DHCP/ALL	Displays SHOW DHCP/SUBNET output for all subnets, plus information about static assignments.
SHOW DHCP/CLIENT_IDENTIFIER=*client-id*	Displays all lease binding and static assignment details for the given client ID.
SHOW DHCP/CONFIGURATION	Writes all configuration and lease information to a dump file.
SHOW DHCP/HARDWARE_ADDRESS= *hardware-address*	Displays all lease binding and static assignment details for the given hardware address.
SHOW DHCP/IP_ADDRESS=*ip-address*	Displays lease binding details for the given IP address. Static assignments are not supported.

Command	**Description**
SHOW DHCP/LEASES	Displays brief information about each lease.
SHOW DHCP/POOLS	Displays address pool availability.
SHOW DHCP/STATUS	Checks if the DHCP server is running.
SHOW DHCP/SUBNET=*ip-address*	Displays brief information about each IP address in the same shared network as the given IP address.
SHOW DHCP/VERIFY	Checks the syntax of the configuration file.
SHOW DHCP/VERSION	Displays the version of the DHCP server.

See the SHOW DHCP, SET DHCP, RELEASE DHCP, REMOVE DHCP, and STOP/DHCP command descriptions in the *NETCU Command Reference*.

DHCP Configuration

You can configure the DHCP server using CNFNET by typing @TCPWARE:CNFNET DHCP.

You can configure the following items:

- Enable or disable the DHCP server.
- Set the debug logging level.
- Set the debug log file name.
- Log debug messages to OPCOM or not.

See the DHCP configuration description in the *Installation and Configuration Guide.*

DHCP Conversion Tool

The DHCP_CONVERSION_TOOL assists in changing from the DHCP server in TCPware V5.3 and earlier to the DHCP server in TCPware V5.4. This tool converts the configuration file and lease file from the 5.3 DHCP server to the format of the configuration and lease files of the 5.4 DHCP server. It is run automatically by the command procedure that starts the DHCP server, TCPWARE:DHCP_CONTROL.COM. However, it is recommended that you run the conversion tool and verify the output ahead of time.

The conversion tool reads the old DHCP configuration file or BOOTP configuration file and writes out a configuration file in the new format. All information from the old configuration file is in the new file. Information that does not have an equivalent in the new configuration file is represented with comment lines.

The conversion tool then reads the old lease file and writes out a lease file in the new format.

The old configuration and lease files are retained in their original location so you can use them to validate the new configuration and lease files.

You can also run the conversion tool directly by specifying the names of the input and output files at the command prompt. If they are not specified, the tool prompts you for them. Enter “NONE” for the file name if you do not have the input file.

For example: if you do not have a lease file, then enter:

```
$ dhcpconvert :== $tcpware:dhcp_conversion_tool.exe
$ dhcpconvert dhcptab. none dhcpd.conf dhcpd.leases
```

Enter a question mark (?) to get help for the tool.

The default file names are:

TCPWARE:DHCPTAB.	The old DHCP configuration file.
TCPWARE:BOOTPTAB.	The old BOOTP configuration file. Read only if the DHCPTAB file does not exist.
TCPWARE:DHCPD.CONF	The new DHCP configuration file.
TCPWARE:DHCP_LEASE.DAT	The old DHCP lease file containing the lease status.
TCPWARE:DHCPD.LEASES	The new DHCP lease file.

Warning! Review the output lease file carefully. The old TCPware lease file does not contain the hardware type, so the conversion tool usually uses ethernet. If you are using a token ring network, change the ’hardware ethernet’ statements to ’hardware token-ring’ statements. The old TCPware lease file also does not contain whether the ID was a hardware address or client identifier, so review carefully all ’uid’ and ’hardware’ statements.

Introducing the Configuration File

TCPware supplies a template DHCP configuration file, TCPWARE:DHCPD_CONF.TEMPLATE, that contains comments and a number of examples to help you enter information for your hosts. Using this template as a guide, create a DHCP configuration file at TCPWARE:DHCPD.CONF (with any text editor) containing the entries you need for your network and hosts.

The dhcpd.conf file is a free-form ASCII text file. The file may contain extra tabs and new lines for formatting purposes and comments may be placed anywhere within the file (except within quotation marks). Comments begin with the # character and end at the end of the line. See Example 2-2. Keywords in the file are case-insensitive.

The file consists of a list of statements specify which fall into two categories: *parameters* and *declarations*.

Parameter statements always specify one of the following:

- How to do something (for example, how long a lease to offer)
- Whether to do something (for example, should the DHCP server provide addresses to unknown clients)
- What parameters to provide to the client (for example, use gateway 220.177.244.7)

Global parameters are at the beginning of the file. Some examples of global parameters are the organization's domain name and the addresses of the name servers (if they are common to the entire organization).

It is legal to specify host addresses in parameters as domain names rather than as numeric IP addresses. If a given *hostname* resolves to more than one IP address (for example, if that host has two ethernet interfaces), both addresses are supplied to the client.

Both the *shared-network* statement and the *subnet* statement can have parameters.

The most obvious reason for having subnet-specific parameters is that each subnet, of necessity, has its own router; for example:

```
option routers 204.254.239.1;
```

Note that the address is specified numerically; this is not required. If you have a different domain name for each interface on your router, it is appropriate to use the domain name for that interface instead of the numeric address. However, there may be only one domain name for all a router's IP addresses, and it would not be appropriate to use that name here.

Parameters starting with the *option* keyword correspond to actual DHCP options. Parameters that do not start with the *option* keyword either control the behavior of the DHCP server (for example, how long a lease the DHCP server will give out), or specify client parameters that are not optional in the DHCP protocol (for example, server-name and filename).

Each host can have host-specific parameters. These could include such things as the:

- Hostname option
- Name of a file to upload (the *filename* parameter)
- Address of the server from which to upload the file (the *next-server* parameter)

In general, any parameter can appear where ever that parameters are allowed, and will be applied according to the scope in which the parameter appears.

All parameters must be specified first before you can specify any declarations that depend on those parameters. Parameters should be set inside declarations so they can be set on a per-subnet or a per-host basis.

Declarations are used to:

- Describe the topology of the network.
- Describe clients on the network.
- Provide addresses that can be assigned to clients.
- Apply a group of parameters to a group of declarations.

Declarations about network topology include the *subnet* and the *shared-network* declarations.

For every subnet to be served, and for every subnet connected to the DHCP server, there must be one *subnet* declaration. This declaration tells the DHCP server how to recognize that an address is on that particular subnet. A *subnet* declaration is required for each subnet even if no addresses is dynamically allocated on that subnet.

There are different declarations required for different situations. The following are the basic declarations in a configuration file.

Declarations	How Used
range	For clients with dynamically assigned addresses, a range declaration must appear within the *subnet* declaration.
host	For clients with statically assigned addresses, or for installations in which only known clients are served, each client must have a host declaration.
group	If parameters are applied to a group of declarations that are not related strictly on a per subnet basis, the group declaration can be used.

Some installations have physical networks allowing more than one IP subnet to operate. For example, if your site has a requirement that 8-bit subnet masks be used, but a department with a single physical ethernet network expands beyond 254 nodes, you may have to run two 8-bit subnets on the same ethernet until a new physical network is added. In this case, you can enclose the *subnet* declarations for the two networks in a *shared-network* declaration.

If you have clients in a department on more than one subnet, it may be best to offer those clients a uniform set of parameters that are different than what would be offered to clients from other departments on the same subnet.

- For clients declared explicitly with *host* declarations, enclose the declarations in a *group* declaration using the parameters that are common to that department.
- For clients with dynamically assigned addresses, group parameter assignments by network topology. Alternately, host declarations can provide parameters and if they have no fixed-address parameter, the clients get an address dynamically assigned. See Example 2-1.

When a client is to be booted, its boot parameters are determined by consulting the:

1	Client's *host* declaration (if any)
2	*Group* declaration (if any) that enclosed the host declaration
3	*User class* declaration (if any) (DHCP only)
4	*Vendor class* declaration (if any) (DHCP only)
5	*Subnet* declaration for the subnet on which the client is booting

6	*Shared-network* declaration (if any) containing that subnet
7	Top-level parameters that may be specified outside of any declaration

When searching for a *host* declaration, the DHCP server looks for one with a fixed-address parameter that matches the subnet or shared network on which the client is booting. If no such entry is found, it looks for an entry with no fixed-address parameter. If no such entry is found again, the DHCP server acts as if there is no entry in the dhcpd.conf file for that client, even if there is an entry for that client on a different subnet or shared network.

Imagine that you have a site with a lot of NCD X-Terminals. These terminals come in a variety of models, and you want to specify the boot files for each model. You could have *host* declarations for each server and group them by model:

Example 2-1 Host Declarations

```
group {
  filename "Xncd19r";
  next-server ncd-booter;
  host ncd1 { hardware ethernet 0:c0:c3:49:2b:57; }
  host ncd4 { hardware ethernet 0:c0:c3:80:fc:32; }
  host ncd8 { hardware ethernet 0:c0:c3:22:46:81; }
}
group {
  filename "Xncd19c";
  next-server ncd-booter;
  host ncd2 { hardware ethernet 0:c0:c3:88:2d:81; }
  host ncd3 { hardware ethernet 0:c0:c3:00:14:11; }
}
group {
  filename "XncdHMX";
  next-server ncd-booter;
  host ncd1 { hardware ethernet 0:c0:c3:11:90:23; }
  host ncd4 { hardware ethernet 0:c0:c3:91:a7:8; }
  host ncd8 { hardware ethernet 0:c0:c3:cc:a:8f; }
}
```

Verifying the DHCP Configuration

After modifying the configuration file, it is good practice to verify the syntax by entering the following NETCU command:

```
$ NETCU SHOW DHCP/VERIFY[/FILENAME=<config-file>] -
_$ [/OUTPUT=<output-file>]
```

This command causes the DHCP server to run enough to read and parse the configuration file. The DHCP server displays a copyright notice and a message for each syntax error encountered. If the DHCP server displays only the copyright notice, the configuration file has no syntax errors.

The /FILENAME qualifier optionally specifies where the configuration file is. If you do not specify the /FILENAME qualifier, the DHCP server reads the default configuration file TCPWARE:DHCPD.CONF. The /OUTPUT qualifier optionally sends command output to the specified file.

Reloading the DHCP Configuration

If you modify TCPWARE:DHCPD.CONF after starting the DHCP server, reload the DHCP configuration by restarting the DHCP server by using @TCPWARE:RESTART DHCP. When the DHCP server restarts, it rereads the configuration file**.**

DNS Dynamic Updates Within DHCP

The DHCP server performs dynamic updates to DNS using DNS's dynamic updating functionality. To be sure that updates are allowed from the DHCP server, see Chapter 1, *Domain Name Services*. The *allow-update { <address_match_list> };* statement in the Zone section enables the DNS server to allow updates from that system.

The following statements in the DHCP server configuration file are related to dynamic updating:

- *allow/deny dynamic-update;*
- *allow/deny update-A-record;*
- *allow/deny name-by-client;*

Dynamic updates can be enabled or disabled by using the *allow/deny dynamic-update* statement in the configuration file. The default is to not perform dynamic updates. Dynamic updates can be turned on or off on a per subnet basis.

Note! Dynamic updates are done for statically assigned IP addresses for DHCP clients, dynamically leased IP addresses for DHCP clients, and dynamically assigned IP addresses for BOOTP clients. The updates are not done for static assignments to BOOTP clients.

When dynamic updating is enabled, the DHCP server determines the client's Fully Qualified Domain Name (FQDN) and assigns it an IP address. The FQDN is determined either by what the client sends or by what is in the configuration file. This behavior is controlled by the *allow/deny name-by-client* statement in the configuration file.

The FQDN and IP address are used in the dynamic update to create a PTR resource record (RR). The DHCP server also optionally creates an A RR. This option is enabled or disabled by using the *allow/deny update-A-record* statement in the configuration file. The default is to not create the A RR. This can be set on a per subnet basis. See Chapter 1, *Domain Name Services* for information about PTR resource records and A resource records.

When dynamic updating is allowed, the DHCP server adds the resource records whenever an IP address is leased to a client. The RRs are deleted if the IP address is released or if the IP address is leased to a different client.

Host Name Generation

Some DHCP clients require that the server send them a host name. The TCPware DHCP server can generate a host name if it cannot get the host name in another way. The generated host name can contain parts of the host's IP address, client ID, and/or MAC address. This host name is sent to the client and is combined with the domain name to create the Fully Qualified Domain Name (FQDN) required for dynamic DNS updates. See the *DNS Dynamic Updates Within DHCP* section. As described in the DNS updates section, the *allow/deny name-by-client* statement in the configuration file controls whether the DHCP server uses information from the client to determine the host name and FQDN.

The DHCP server generates a host name if it is enabled to do so and either of the following is specified:

- *allow name-by-client* and the client does not send a host name
- *deny name-by-client* and the DHCP server does not find a host name in the configuration file or in DNS (if *get-lease-hostnames* is set)

To enable the DHCP server to generate host names, specify in the configuration file an *option host-name* statement with a value containing certain key values in addition to any characters that are valid for the *host-name* option (see Table 2-2). The *option host-name* statement can be specified for example at the top level in a *subnet* statement, or in a *host* statement.

The key values are as follows. You can include more than one in the same host-name value. Note that some of these do not by themselves generate a unique identifier.

Key	**Meaning**
%A	First byte of the host's IP address. Example: for address 10.24.25.201, the key would return 10.
%B	Second byte of the host's IP address. Example: for address10.24.25.201, the key would return 24.
%C	Third byte of the host's IP address. Example: for address 10.24.25.201, the key would return 25.
%D	Fourth byte of the host's IP address. Example: for address 10.24.25.201, the key would return 201.
%H	Host part of the host's IP address. Example: for address 10.24.25.201 with subnet mask 255.255.0.0, the key would return 6601.
%I	Client Identifier sent by the host.
%-I	Client ID as above, except that hyphens (-) are used to separate each byte.
%M	MAC address of the host.
%-M	MAC address of the host, as above, except that hyphens (-) are used to separate each byte.

Key	Meaning
%N	Host name sent by the client, if any. If none, “Host”.
%S	Subnet part of the host's IP address. Example: for address 10.24.25.201 with subnet mask 255.255.0.0, the key would return 102400.
%-S	Subnet part of the host's IP address, as above, except that hyphens (-) are used to separate each byte. For example: 10-24-0-0.

You can intersperse string constants such as hyphens between key definitions. However, if the generated host name exceeds 63 characters, it is truncated. Here is an example host-name statement:

```
option host-name "Host%H-%-S";
```

For a lease pool defined with an address range of 192.168.11.6 through 192.168.11.10 and a subnet mask of 255.255.255.0, the DHCP server generates the following host names:

```
Host6-192-168-11-0
Host7-192-168-11-0
Host8-192-168-11-0
Host9-192-168-11-0
Host10-192-168-11-0
```

The %N key allows you to use the host name as sent by the client (option 12) and then add something unique to it to generate a unique name. For example, if multiple clients all send the name "dilbert" you can make them unique by appending the MAC (hardware) address, as follows:

```
deny name-by-client;
option host-name "%N-%M";
```

This would generate the host name "dilbert-010203040506" for a client with hardware address 01:02:03:04:05:06.

Configuration File Declarations and Parameters

Table 2-1 describes the declarations and parameters you can use in a configuration file.

See Table 2-8 for a list of DHCP Safe-failover-related configuration file statements.

Table 2-1 DHCP Statements

Statement	Description
allow and deny	Use the *allow* and *deny* statements to control the behavior of the DHCP server. Use the *unknown-clients* flag to tell the DHCP server to dynamically assign addresses to unknown clients or to not assign addresses to unknown clients. The default is to *allow* dynamic address assignments to unknown clients. Use the *bootp* flag to tell the DHCP server to respond to bootp queries or to not respond to bootp queries. The default is to *allow* bootp queries. Use the *booting* flag to tell the DHCP server to respond to queries from a particular client or to not respond to queries from a particular client. This keyword only works when it appears in a host declaration. The default is to *allow* booting. If it is disabled for a particular client, that client will not be able to get an address from the DHCP server. Use the *dynamic-update* flag to tell the DHCP server to perform dynamic DNS updates or to not perform them. The default is to *deny* dynamic DNS updates. Use the *name-by-client* flag to tell the DHCP server to determine the hostname and Fully Qualified Domain Name (FQDN) for dynamic DNS updates from information sent by the client or from information in the configuration file. The default is to *deny* use of client-specified information. Use the *dhcpinform* flag to tell the DHCP server to respond to DHCPINFORM messages or to not respond. The default is to *allow* DHCPINFORM messages.

Table 2-1 DHCP Statements (Continued)

Statement	**Description**
allow and deny (continued)	Use the *update-A-record* flag to tell the DHCP server to update the A resource record or not when performing DNS updates (the PTR resource record is always updated). The default is to *deny* updating the A resource record. Use the *ras-servers* flag to tell the DHCP server to respond to queries from Microsoft (R) NT RAS Servers or to not respond to NT RAS queries. The default is to allow NT RAS queries. allow unknown-clients; allow dynamic-update; deny unknown-clients; deny dynamic-update; allow bootp; allow name-by-client; deny bootp; deny name-by-client; allow booting; allow update-A-record; deny booting; deny update-A-record allow dhcpinform; allow ras-servers; deny dhcpinform; deny ras-servers;
always-reply-rfc1048	Some BOOTP clients expect RFC 1048-style responses, but do not follow RFC 1048 rules when sending their requests. You can determine if a client is having this problem: • if it is not getting the options you have configured for it. and • if you see in the server log the message "(non-rfc1048)" printed with each BOOTREQUEST that is logged. If you want to send RFC 1048 options to this kind of client, set the *always-reply-rfc1048* option in that client's host declaration. The DHCP server responds with an RFC 1048-style vendor options field. This flag can be set in any scope, and affects all clients covered by that scope. `always-reply-rfc1048 flag;`

Table 2-1 DHCP Statements (Continued)

Statement	**Description**
authoritative; not authoritative;	The DHCP server normally assumes the configuration information about a given network segment is correct and is authoritative. So if a client requests an IP address on a given network segment that the server knows is not valid, the server responds with a DHCPNAK message, causing the client to forget its IP address and try to get a new one. If a DHCP server is being configured by someone other than the network administrator and who does not want to assert this level of authority, the statement *not authoritative* should be written in the appropriate scope in the configuration file. Usually, writing *not authoritative* at the top level of the file should be sufficient. However, if a DHCP server is to be set up so that it is aware of some networks for which it is authoritative and some networks for which it is not, it may be more appropriate to declare authority on a per-network-segment basis. The most specific scope for which the concept of authority makes sense is the physical network segment - either a shared-network statement or a subnet statement that is not contained within a shared-network statement. It is not meaningful to specify that the server is authoritative for some subnets within a shared network, but not authoritative for others, nor is it meaningful to specify that the server is authoritative for some host declarations and not others. ***Note!*** For network segments for which the DHCP server is not authoritative, you may want to specify *deny dhcpinform*.
default-lease-time	*Time* is the length (in seconds) that the DHCP server assigns to a lease if the requesting client did not ask for a specific amount of time for the lease to be active. The infinite lease value is 0. The default is 12 hours. You should set the value of *default-lease-time* no larger than the value of *max-lease-time*. `default-lease-time time;`

Table 2-1 DHCP Statements (Continued)

Statement	Description
dynamic-bootp-lease-cutoff	Use the *dynamic-bootp-lease-cutoff* statement to set the ending time for all leases dynamically assigned to BOOTP clients. By default, the DHCP server assigns infinite leases to all BOOTP clients because they do not have any way of renewing leases, and do not know that their leases could expire. However, it may make sense to set a cutoff date for all BOOTP leases. For example, the end of a school term, or the time at night when a facility is closed and all machines are required to be powered off. *Date* should be the date all assigned BOOTP leases will end. The date is specified in the form: **W YYYY/MM/DD HH:MM:SS** where: **W** is the day of the week, from zero (Sunday) to six (Saturday). **YYYY** is the year, including the century. **MM** is the number of the month, from 01 to 12. **DD** is the day of the month, counting from 01. **HH** is the hour, from 00 to 23. **MM** is the minute, from 00 to 59. **SS** is the second, from 00 to 59. The time is always in Greenwich Mean Time, not local time. `dynamic-bootp-lease-cutoff date;`
dynamic-bootp-lease-length	Use the *dynamic-bootp-lease-length* statement to set the length of leases dynamically assigned to BOOTP clients. You may be able to assume that a lease is no longer in use if its holder has not used BOOTP or DHCP to get its address within a certain time period. The length of the time period is your judgment call. Specify *length* in seconds. The infinite lease value is 0. If a BOOTP client reboots during a timeout period, the lease duration is reset to *length* so a BOOTP client that boots frequently never loses its lease. This parameter should be adjusted with extreme caution. The default is an infinite lease. `dynamic-boot-lease-length length;`
filename	Use the *filename* statement to specify the name of the initial boot file that is to be loaded by a client. The *filename* should be recognizable to whatever file transfer protocol the client can be expected to use. `filename filename;`

Table 2-1 DHCP Statements (Continued)

Statement	**Description**
fixed-address	Use the *fixed-address* statement to assign one or more fixed IP addresses to a client. It should only appear in a *host* declaration. With more than one address supplied, the client is assigned the address that corresponds to the network on which it is booting. If none of the supplied addresses are on the network the client is booting from, the client will not match the host declaration containing that *fixed-address* statement. Each address should be either an IP address or a domain name that resolves to one or more IP addresses. `fixed-address address [,address];`
get-lease-hostnames	Use the *get-lease-hostnames* statement to tell the DHCP server to look up the domain name corresponding to each address in the lease pool and use that address for the DHCP hostname option. If *flag* is true, the lookup is done for all addresses in the current scope. If *flag* is false (the default), lookups are not done. `get-lease-hostnames flag;`
group	Use the *group* statement to apply one or more parameters to a group of declarations. You can use it to group hosts, shared networks, subnets, or other groups. `group {[parameters] [declarations]}`
hardware	Use the *hardware* clause inside a *host* statement to specify the network hardware address of a BOOTP or DHCP client. *hardware-type* must be the name of a physical hardware interface type. Ethernet, Token-Ring, and FDDI are the only recognized types. The *hardware-address* should be a set of hexadecimal octets (numbers from 0 through ff) separated by colons (:). `hardware hardware-type hardware-address;`

Table 2-1 DHCP Statements (Continued)

Statement	Description
host	You may specify *host* statements for BOOTP or DHCP clients, although this is not required unless booting is only enabled for known hosts, or, if you want to give the BOOTP or DHCP client a fixed address. *hostname* should be a name identifying the host. *hostname* is required. If you want to boot a DHCP or a BOOTP client on more than one subnet with fixed addresses, you can either specify: • More than one address in the *fixed-address* parameter, or • More than one *host* statement. You should use multiple *host* statements if client-specific boot parameters must change because of the network the client is attached to. If you want to boot a client using a fixed address on one subnet, but the client should be allocated a dynamic address on another subnet, then specify a *host* statement without a *fixed-address* clause in addition to the *host* statement with the fixed address for the first subnet. A *Host* declaration is matched to actual DHCP or BOOTP clients by matching the *dhcp-client-identifier* option specified in the *host* declaration to the one supplied by the client. If the *host* declaration or the client does not provide a *dhcp-client-identifier* option, the *hardware* parameter in the *host* declaration is matched to the network hardware address supplied by the client. Since BOOTP clients do not normally provide a *dhcp-client-identifier,* the hardware address must be used for all *host* statements for clients using the BOOTP protocol. `host hostname {[parameters][declarations]}`
max-lease-time	*Time* is the maximum amount of time (in seconds) a lease is allowed to be. The infinite lease value is 0. The default is 24 hours. You should set the value of *max-lease-time* at least as large as *default-lease-time*. `max-lease-time time;`

Table 2-1 DHCP Statements (Continued)

Statement	Description
next-server	Use the *next-server* statement to specify the host address of the server from where the client will load the initial boot file (specified in the *filename* statement). *server-name* should be a numeric IP address or a domain name. The DHCP server's IP address is used if no *next-server* parameter applies to a given client. `next-server name;`
option	This statement specifies actual DHCP protocol options to send to the client. The option statement is described in the next section.
ping	The DHCP server uses ping to check if a particular IP address is in use by sending a packet of information and waiting for a response. This statement turns ping on and off. The default is on. `ping flag;`
ping-retries	This statement defines the number of times the DHCP server pings an IP address before it concludes that the address is not in use. The default is 1. `ping-retries count;`
ping-timeout	This statement defines the time (in seconds) that ping should wait for a response. The default is 1 second. `ping-timeout time;`
range	There must be at least one *range* statement for any subnet with addresses that can be dynamically assigned. The range statement gives the lowest and the highest IP addresses available in a range. All IP addresses in the range should be in the subnet where the *range* statement is declared. The *dynamic-bootp* flag may be specified if addresses in the selected range can be dynamically assigned to BOOTP clients as well as to DHCP clients. Omit *high-address* when specifying a single address. `range [dynamic-bootp] low-address [high-address];`

Table 2-1 DHCP Statements (Continued)

<table>
<tr><th>Statement</th><th>Description</th></tr>
<tr><td>requested-options-only <u>flag</u>;</td><td>Some DHCP clients cannot handle receiving any options other than the ones they request or a specific list of options. In order to send just the options requested by the client, specify "requested-options-only true;" in the configuration for that client. In order to send a specific set of options, you can set requested-options-only to true and specify the dhcp-parameter-request-list option. For example, the following sends only the subnet-mask, routers, and domain-name-servers options to the client (assuming they are defined in the configuration file):
<pre>host restricted {
    hardware ethernet 01:02:03:04:05:06;
    option dhcp-parameter-request-list 1, 3, 6;
    requested-options-only true;
}</pre></td></tr>
<tr><td>server-identifier</td><td>The server-identifier statement can be used to define the value that is sent in the DHCP Server Identifier option. The value specified MUST be an IP address for the DHCP server, and must be reachable by all clients which it is sent to.

The use of the server-identifier statement is not recommended. The only reason to use it is to force a value other than the default value to be sent on occasions where the default value would be incorrect. The default value is the first IP address associated with the physical network interface on which the request arrived.

The usual case where the server-identifier statement needs to be sent is when a physical interface has more than one IP address, and the one being sent by default is not appropriate for some or all clients served by that interface.

Another case is when an alias is defined for the purpose of having a consistent IP address for the DHCP server, and it is desired that the clients use this IP address when contacting the server.

Supplying a value for the dhcp-server-identifier option is equivalent to using the server-identifier statement.

<code>server-identifier <u>hostname</u>;</code></td></tr>
<tr><td>server-name</td><td>Use the server-name statement to inform the client of the server’s name from which it is booting. name should be the name provided to the client.

<code>server-name <u>name</u>;</code></td></tr>
</table>

Table 2-1 DHCP Statements (Continued)

Statement	Description
shared-network	Use this statement to inform the DHCP server that some IP subnets share the same physical network. Declare all subnets in the same shared network within a *shared-network* statement. *Parameters* specified in the shared-network statement will be used when booting clients on those subnets unless parameters provided at the subnet or host level override them. If more than one subnet in a shared network has addresses available for dynamic allocation, those addresses are collected into a common pool. There is no way to specify which subnet of a shared network a client should boot on. *Name* should be the name of the shared network. Make the name descriptive as it will be used when printing debugging messages. Give it a syntax of a valid domain name (although it will never be used as such), or any arbitrary name enclosed in quotation marks. `shared-network name {[parameters] [declarations]}`

Table 2-1 DHCP Statements (Continued)

<table>
<tr><th>Statement</th><th>Description</th></tr>
<tr><td>subnet</td><td>This statement contains information specific to a subnet. The information communicates the following to DHCP:
Enough information for DHCP to determine if an IP address is on that subnet.
What the subnet-specific parameters are.
What addresses may be dynamically allocated to clients booting on that subnet.
Use the range declaration to specify what addresses are available to be dynamically allocated to clients booting on the subnet.

Two things are required to define a subnet:
The subnet-number
The netmask
The subnet-number and the netmask entry is an IP address or domain name that resolves to the subnet-number or the netmask of the subnet being described. The subnet-number and the netmask are enough to determine if any given IP address is on the specified subnet.

Note! A netmask must be given with every subnet declaration. If there is any variance in subnet masks at a site, use a subnet-mask option statement in each subnet declaration to set the desired subnet mask. The subnet-mask option statement overrides the subnet mask declared in the subnet statement.

<code>subnet <u>subnet-number</u> netmask
<u>netmask</u> {[<u>parameters</u>][<u>declarations</u>]}</code></td></tr>
</table>

Table 2-1 DHCP Statements (Continued)

Statement	Description

use-host-decl-names	If the *use-host-decl-names* parameter is true, the name provided for each host declaration is given to the client as its hostname. The default is false. For example,
	`group {`
	`use-host-decl-names on;`
	`host joe {`
	`hardware ethernet 08:00:2b:4c:29:32;`
	`fixed-address joe.fugue.com;`
	`}`
	`}`
	is equivalent to
	`host joe {`
	`hardware ethernet 08:00:2b:4c:29:32;`
	`fixed-address joe.fugue.com;`
	`option host-name “joe”;`
	`}`
	An *option host-name* statement within a host declaration overrides the use of the name in the host declaration.
	`use-host-decl-names flag;`

use-lease-addr-for-default-route	If the *use-lease-addr-for-default-route* parameter is true in a given scope, the IP address of the lease being assigned is sent to the client instead of the value specified in the routers option (or sending no value at all). This causes some clients to ARP for all IP addresses, which can be helpful if your router is configured for proxy ARP. If *use-lease-addr-for-default-route* is enabled and an *option routers* statement are both in scope, the routers option is preferred. `use-lease-addr-for-default-route flag;`

user-class	Use this statement to specify parameters and options for clients with a particular user class. User class is sent by the client in the User Class Information option. (DHCP only)
	`user-class “user-class-identifier”`
	`{[parameters] [declarations]}`

Table 2-1 DHCP Statements (Continued)

Statement	Description

vendor-class	Use this statement to specify parameters and options for clients with a particular vendor class. Vendor class is sent by the client in the Vendor Class Identifier option. (DHCP only)
	
	`vendor-class "vendor-class-identifier"`
	`{[parameters] [declarations]}`

DHCP Options

The Dynamic Host Configuration protocol allows the client to receive options from the DHCP server describing the network configuration and various services that are available on the network. When configuring the DHCP server, options must often be declared. The syntax for declaring options, and the names and formats of the options that can be declared, are in Table 2-2.

DHCP option statements always start with the keyword *option*, followed by an option name, which is followed by option data. Only options needed by clients must be specified.

Option data comes in these formats:

- The **ip-address** data type can be entered either as an explicit IP address (for example., 239.254.197.10) or as a domain name (for example., haagen.isc.org). When entering a domain name, be sure that the domain name resolves to the single IP address.
- The **int32** and **uint32** data types specify signed and unsigned 32-bit integers. The **int16** and **uint16** data types specify signed and unsigned 16-bit integers. The **int8** and **uint8** data types specify signed and unsigned 8-bit integers. Unsigned 8-bit integers are also sometimes referred to as octets.
- The **string** data type specifies an NVT ASCII string. It must be enclosed in quotation marks. For example, *option domain-name "isc.org";*
- The **flag** data type specifies a boolean value. Booleans can be either true (ON) or false (OFF). You can use TRUE and FALSE, or ON and OFF.
- The **data-string** data type specifies either an NVT ASCII string enclosed in quotation marks, or a series of octets specified in hexadecimal format, separated by colons. For example, *option dhcp-client-identifier "CLIENT-FOO";* or *option dhcp-client-identifier 43:4c:49:54:2d:46:4f:4f;*

If the option value is a list (such as for the routes option), you must list them in the configuration file in the order you want the client to use the values. The DHCP server does not re-order them.

Table 2-2 describes the standard DHCP options. Underlined items indicate user input items.

Table 2-2 DHCP Options

Option	Description
option all-subnets-local flag;	Use this option to indicate whether or not to assume all subnets of the client's IP network use the same MTU as the client's subnet. ON means assume all subnets share the same MTU. OFF means assume some subnets have smaller MTUs.
option arp-cache-timeout uint32;	Use this option to identify the timeout (in seconds) for ARP cache entries.
option bootfile-name string;	Use this option to identify a bootstrap file. If this option is supported by the client, it should have the same effect as the *filename* declaration. BOOTP clients are unlikely to support this option. Some DHCP clients support it; others require it.
option boot-size uint16;	Use this option to specify the length in 512-octet blocks of the client's default boot image.
option broadcast-address ip-address;	Use this option to identify the broadcast address in use on the client's subnet. See STD 3 (RFC 1122), section 3.2.1.3 for legal values for broadcast addresses.
option cookie-servers ip-address [, ip-address ...];	Use this option to list RFC 865 cookie servers in order of preference.
option default-ip-ttl uint8;	Use this option to identify the default time-to-live the client should use on outgoing datagrams.
option default-tcp-ttl uint8;	Use this option to identify the default TTL to use when sending TCP segments. The minimum value is 1.
option dhcp-client-identifier data-string;	Use this option to specify a DHCP client identifier only in a *host* declaration. The DHCP server uses it to locate the *host* record by matching against the client identifier.

Table 2-2 DHCP Options (Continued)

Option	Description
option dhcp-max-message-size uint16;	Use this option to specify the maximum length DHCP message that the client is able to accept. Use this option in the DHCP configuration file to supply a value when the client does not. ***Note!*** Use this option with caution. Make sure that the client can accept a message of the specified size.
option dhcp-parameter-request-list uint8[,uint8...];	Use this option to request that the server return certain options. Use this option in the DHCP configuration file to override the client's list, or to supply a list when the client does not. The value is a list of valid DHCP option codes as listed in RFC 2132.
option dhcp-server-identifier ip-address;	Use this option to identify the value sent in the DHCP Server Identifier option. The value must be an IP address for the DHCP server, and must be reachable by all clients it is sent to. It is recommended to NOT use the dhcp-server-identifier option. The only reason to use it is to force a value other than the default value to be sent on occasions where the default value would be incorrect. The default value is the first IP address associated with the physical network interface on which the request arrived. The usual case where the dhcp-server-identifier option needs to be sent is when a physical interface has more than one IP address, and the one being sent by default is not appropriate for some or all clients served by that interface.
option domain-name-servers ip-address [, ip-address ...];	Use this option to list Domain Name System (STD 12, RFC 1035) name servers in order of preference.
option domain-name string;	Use this option to identify the domain name the client should use when resolving hostnames via the Domain Name System.
option extensions-path string;	Use this option to indicate the path-name of a file the client should load containing more options.
option finger-server ip-address [, ip-address ...];	Use this option to list the Finger servers in order of preference.

Table 2-2 DHCP Options (Continued)

Option	**Description**
option font-servers <u>ip-address</u> [, <u>ip-address</u> ...];	Use this option to list X Window System Font servers in order of preference.
option host-name <u>string</u>;	Use this option to name the client. The name may or may not be qualified with the local domain name. It is preferable to use the domain-name option to specify the domain name. See RFC 1035 for character set restrictions. The host-name option is also used to specify a template for hostname generation. See the *Host Name Generation* section.
option ieee802-3-encapsulation <u>flag</u>;	If the interface is an Ethernet, use this option to indicate whether the client uses Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation. OFF means use RFC 894 encapsulation. ON means use RFC 1042 encapsulation.
option ien116-name-servers <u>ip-address</u> [, <u>ip-address</u> ...];	Use this option to list IEN 116 name servers in order of preference.
option impress-servers <u>ip-address</u> [, <u>ip-address</u> ...];	Use this option to list Imagen Impress servers in order of preference.
option interface-mtu <u>uint16</u>;	Use this option to identify what MTU value to use on this interface. The minimum legal value is 68.
option ip-forwarding <u>flag</u>;	Use this option to indicate if the client should configure its IP layer for packet forwarding. ON means disable forwarding. OFF means enable forwarding.
option irc-server <u>ip-address</u> [, <u>ip-address</u> ...];	Use this option to list the IRC servers in order of preference.
option log-servers <u>ip-address</u> [, <u>ip-address</u> ...];	Use this option to list MIT-LCS UDP log servers in order of preference.
option lpr-servers <u>ip-address</u> [, <u>ip-address</u> ...];	Use this option to list RFC 1179 line printer servers in order of preference.
option mask-supplier <u>flag</u>;	Use this option to indicate whether or not the client should respond to subnet mask requests using ICMP. ON means do not respond to subnet mask requests. OFF means respond to subnet mask requests.

Table 2-2 DHCP Options (Continued)

Option	Description
option max-dgram-reassembly uint16;	Use this option to indicate the maximum size datagram the client should be prepared to reassemble. The minimum legal value is 576.
option merit-dump string;	Use this option to indicate the path-name of a file to which the client's core image should be dumped in the event of a client crash. The path is formatted as a character string using the NVT ASCII character set.
option mobile-ip-home-agent ip-address [, ip-address ...];	Use this option to list mobile IP home agents in order of preference. Usually there will be only one agent.
option nds-context data-string;	Use this option to identify the initial NDS context the client should use.
option nds-servers ip-address [, ip-address...];	Use this option to list Novell Directory Services servers in order of preference.
option nds-tree-name data-string;	Use this option to name the NDS tree the client will be contacting.
option netbios-dd-server ip-address [, ip-address ...];	Use this option to list RFC 1001/1002 NetBIOS Datagram Distribution servers in order of preference.
option netbios-name-servers ip-address [, ip-address ...];	Use this option to list RFC 1001/1002 NetBIOS Name Server name servers in order of preference. NetBIOS Name Service is currently more commonly referred to as WINS. WINS servers can be specified using the netbios-name-servers option.
option netbios-node-type uint8;	Use this option to configure configurable NetBIOS over TCP/IP clients as described in RFC 1001/1002. The value is a single octet identifying the client type. Possible node types are 1 B-node: Broadcast--No WINS 2 P-node: Peer--WINS only 4 M-node: Mixed--Broadcast, then WINS 8 H-node: Hybrid--WINS, then Broadcast
option netbios-scope string;	Use this option to specify the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002. See RFC1001, RFC1002, and RFC1035 for character-set restrictions.

Table 2-2 DHCP Options (Continued)

Option	Description
option nis-domain string;	Use this option to specify the client's NIS (Sun Network Information Services) domain. Use the NVT ASCII character set to define the domain character string.
option nis-servers ip-address [, ip-address ...];	Use this option to list NIS servers in order of preference.
option nisplus-domain string;	Use this option to specify the client's NIS+ domain. Use the NVT ASCII character set to define the domain character string.
option nisplus-servers ip-address [, ip-address ...];	Use this option to list NIS+ servers in order of preference.
option non-local-source-routing flag;	Use this option to indicate if the client should configure its IP layer to allow forwarding of datagrams with non-local source routes. ON means disable forwarding. OFF means enable forwarding.
option nntp-server ip-address [, ip-address ...];	Use this option to list NNTP servers in order of preference.
option ntp-servers ip-address [, ip-address ...];	Use this option to list NTP (RFC 1035) servers in order of preference.
option option-nnn data-string;	Use this option to identify any DHCP option not listed here. nnn is the number of the option.
option path-mtu-aging-timeout uint32;	Use this option to specify the timeout to use (in seconds) when aging Path MTU values that were discovered by the mechanism defined in RFC 1191.
option path-mtu-plateau-table uint16 [, uint16 ...];	Use this option to specify a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is a list of 16-bit unsigned integers. You must list them in order from smallest to largest. The minimum MTU value cannot be smaller than 68.
option perform-mask-discovery flag;	Use this option to indicate whether or not the client should perform subnet mask discovery using ICMP. ON means do not perform mask discovery. OFF means perform mask discovery.

Table 2-2 DHCP Options (Continued)

Option	**Description**
option policy-filter ip-address ip-address [, ip-address ip-address ...];	Use this option to indicate the policy filters for non-local source routing. The filters consist of IP addresses and masks that indicate which destination/ mask pairs to use when filtering incoming source routes. The client should discard any source routed datagram whose next-hop address does not match one of the filters. See STD 3 (RFC 1122) for more information.
option pop-server ip-address [, ip-address ...];	Use this option to list POP3 servers in order of preference.
option resource-location-servers ip-address [, ip-address ...];	Use this option to list RFC 887 Resource Location servers in order of preference.
option root-path string;	Use this option to specify the path-name that contains the client's root disk. The path is formatted using the NVT ASCII character set.
option router-discovery flag;	Use this option to indicate whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256. ON means do not perform router discovery. OFF means perform router discovery.
option routers ip-address [, ip-address ...];	Use this option to list IP addresses for routers in order of preference.
option router-solicitation-address ip-address;	Use this option to identify the address where the client transmits router solicitation requests.
option smtp-server ip-address [, ip-address ...];	Use this option to list SMTP servers in order of preference.

Table 2-2 DHCP Options (Continued)

Option	**Description**
option static-routes ip-address ip-address [, ip-address ip-address ...];	Use this option to specify a list of static routes that the client should install in its routing cache. If there are multiple routes to the same destination, you should list them in descending order of priority. The routes are made up of IP address pairs. The first address is the destination address; the second address is the router for the destination. The default route (0.0.0.0) is an illegal destination for a static route. Use the *routers* option to specify the default route.
option streettalk-directory-assistance-server ip-address [, ip-address ...];	Use this option to list the StreetTalk Directory Assistance (STDA) servers in order of preference.
option streettalk-server ip-address [, ip-address ...];	Use this option to list the StreetTalk servers in order of preference.
option subnet-mask ip-address;	Use this option indicate the client's subnet mask as per RFC 950. If no subnet mask option is in scope, the DHCP server uses the subnet mask from the subnet declaration on which the address is being assigned. If a subnet mask option is in scope for the address being assigned, it overrides the subnet mask specified in the subnet declaration.
option swap-server ip-address;	Use this option to identify the IP address of the client's swap server.
option tcp-keepalive-garbage flag;	Use this option to indicate whether the client sends TCP keepalive messages with an octet of garbage for compatibility with older implementations. ON means do not send a garbage octet. OFF means send a garbage octet.
option tcp-keepalive-interval uint32;	Use this option to indicate the interval (in seconds) the client TCP waits before sending a keepalive message on a TCP connection. The time is specified as a 32-bit unsigned integer. 0 (zero) means do not generate keepalive messages unless requested by an application.

Table 2-2 DHCP Options (Continued)

Option	Description
option tftp-server-name string;	Use this option to identify a TFTP server. If this option is supported by the client, it should have the same effect as the *server-name* declaration. BOOTP clients are unlikely to support this option. Some DHCP clients support it; others require it.
option time-offset int32;	Use this option to specify the offset of the client's subnet (in seconds) from Coordinated Universal Time (UTC). Use negative numbers for West of UTC and positive numbers for East of UTC.
option time-servers ip-address [, ip-address ...];	Use this option to list RFC 868 time servers in order of preference.
option trailer-encapsulation flag;	Use this option to indicate if the client negotiates the use of trailers (RFC 893) when using the ARP protocol. ON means do not use trailers. OFF means use trailers.
option vendor-encapsulated-options data-string;	Use this option to specify vendor specific information.
option www-server ip-address [, ip-address ...];	Use this option to list WWW servers in order of preference.
option x-display-manager ip-address [, ip-address ...];	Use this option to list the systems running X Window System Display Manager in order of preference.

DHCP Lease Format

The DHCP server keeps a persistent database of leases it has assigned. This database is a free-form ASCII file containing a series of lease declarations. Every time a lease is acquired, renewed, or released, its new value is recorded at the end of the lease file. So, if more than one declaration appears for a given lease, the last one in the file is the current one.

Currently, the only declaration that is used in the dhcpd.leases file is the *lease* declaration.

```
lease ip-address {statements...}
```

Each lease declaration includes the client's leased IP address. The statements within the braces define the duration of the lease and to whom it is assigned.

Table 2-3 describes the statements the DHCP server puts into a lease file.

See Table 2-8 for a list of DHCP Safe-failover related lease file statements.

Table 2-3 DHCP Lease File Statements

Lease Statement	Description
abandoned;	Records that the DHCP server saw the IP address in use on the network when it was thought to be free. The DHCP server detects active addresses with ping tests or "DHCP decline" messages from DHCP clients.
client-hostname “hostname”;	Records the hostname if the client sends a hostname using the hostname option.
domain-name “domain-name”;	Specifies the DNS domain name sent to the client (if any).
dynamic-bootp;	Indicates the address was leased to a BOOTP client.
ends date;	Records the end time of a lease. Lease dates are specified by the DHCP server as follows: **W YYYY/MM/DD HH:MM:SS** where: **W** is the day of the week, from zero (Sunday) to six (Saturday). **YYYY** is the year, including the century. **MM** is the number of the month, from 01 to 12. **DD** is the day of the month, counting from 01. **HH** is the hour, from 00 to 23. **MM** is the minute, from 00 to 59. **SS** is the second, from 00 to 59. The time is always in Greenwich Mean Time, not local time.
FQDN “fully-qualified-domain-name”;	Specifies the fully qualified domain name used by the DHCP server to perform the dynamic DNS update for the lease (if any).
hardware hardware-type mac-address;	Specifies the hardware type and the MAC address as a series of hexadecimal octets, separated by colons.
hostname “hostname”;	Records the hostname if the DHCP server looks up the hostname in DNS. This happens only if the parameter *get-lease-hostnames* was set.
starts date;	Records the start time of a lease.

Table 2-3 DHCP Lease File Statements (Continued)

Lease Statement	Description
uid client-identifier;	Records the client identifier as a series of hexadecimal octets, regardless of whether the client specifies an ASCII string or uses the hardware type/MAC address format. If the client used a client identifier to acquire its address, the client identifier is recorded using the uid statement.

Working with DHCP Leases

The DHCP server requires that a lease database be present before it will start. Before starting the DHCP server for the first time, make sure there is a TCPWARE:DHCPD.LEASES file. If it doesn't exist, create an empty one.

In order to prevent the lease database from growing without bound, the file is rewritten from time to time. First, a temporary lease database is created and all known leases are dumped to it. Then, the old lease database is renamed TCPWARE:DHCPD.LEASES_OLD. Finally, the newly written lease database is moved into place.

Note! Be aware of the following situation: If the DHCP server process is stopped or the system crashes after the old lease database has been renamed but before the new lease database has been moved into place, the TCPWARE:DHCPD.LEASES file is deleted. The DHCP server will not start. Do not create a new lease file when this happens. If you do, you will lose all your old bindings. Instead, rename TCPWARE:DHCPD.LEASES_OLD to TCPWARE:DHCPD.LEASES, restoring the old, valid lease file, and then start the DHCP server. This guarantees that a valid lease file will be restored.

Abandoned Leases

Abandoned leases are reclaimed automatically. When a client asks for a new address, and the server finds that there are no addresses available, it looks for any abandoned leases. The server allocates the oldest abandoned lease. The standard procedures for checking for lease address conflicts are still followed, so if the abandoned lease's IP address is still in use, it is reabandoned.

If a client requests an abandoned address, the server assumes that the address was abandoned because the lease file was corrupted, and that the client is the machine that responded when the lease was pinged, causing it to be abandoned. In that case, the address is immediately assigned to the requesting client.

Address Lease States in DHCP Dump Files

After obtaining a DHCP dump using NETCU SHOW DHCP/CONFIGURATION, you will see fields in the dump preceded by a pound sign (#). Those fields are values created while the protocol is running and do not directly correlate with the DHCP options listed in Table 2-2. These fields are provided to help you troubleshoot problems. The lease states (denoted by # *State*= in the dump) are described in Table 2-4.

Table 2-4 DHCP Address Lease States

State	Description
Abandoned	The address was seen in use on the network when it was thought to be free. The DHCP server detects active addresses with ping tests or "DHCP decline" messages from DHCP clients.
Bound	The address was assigned in response to a DHCP Request message. If the address lease expires, it remains in "bound" state to help the client regain the same IP address it previously used. The address is actually free. The "(expired)" modifier on the state value indicates this state.
Free	The address has never been bound, and is available in the pool.
Offered	The address has been offered to a client in response to the client's DHCP Discover message, but the client has not asked for the address via a DHCP Request message.
Pinging	The address is in the middle of a ping test.
Static Assignment	The client identifier or hardware address is statically assigned; the binding does not expire.
Reserved for Secondary	Used for DHCP Safe-failover: The address is set aside for the secondary server's emergency use.

Sample DHCPD.CONF File

Example 2-2 shows a sample DHCPD.CONF file.

Example 2-2 Sample DHCPD.CONF File

```
#
# TCPWARE:DHCPD.CONF -- sample DHCP configuration file
#

# option definitions common to all supported networks...
option domain-name “fugue.com”;
option domain-name-servers toccato.fugue.com;
default-lease-time 43200;
option subnet-mask 255.255.255.0;
option time-offset 18000;
use-host-decl-names on;

# Shared network declaration is used to group subnets which share the same
# physical network together. The name is specified so that the shared
```

```
# network can be referred to in log messages --
# it serves no other function.
#
# Note: You must have a subnet declaration for the subnet that the DHCP
# server system is on even if you don't want any address pool for the same
# subnet (or multiple subnets if the system is multi-homed).

shared-network FUGUE  {

# option definitions common to this shared network.
   option subnet-mask 255.255.255.224;
   default-lease-time 600;
   max-lease-time 7200;

# One of the two IP subnets that share this physical network
#
# Address ranges can be specified for each subnet attached to a shared
# network. Since these subnets share the same physical network, addresses
# are pooled together, and assignments are made without regard to the
# actual subnet. If the optional dynamic-bootp keyword is given in the
# address range declaration, then addresses in that range can be assigned
# either with the DHCP protocol or the BOOTP protocol; otherwise, only
# DHCP clients will have addresses allocated from the address range.
#
# Note that each IP subnet can have its own options specific to that
# subnet. Options that are not specified in the subnet are taken from the
# shared network (if any) and then from the global option list.

   subnet 204.254.239.0 netmask 255.255.255.224 {
       range 204.254.239.10 204.254.239.20;
       option broadcast-address 204.254.239.20;
       option routers prelude.fugue.com;
}

# The other subnet that shares this physical network
   subnet 204.254.239.32 netmask 255.255.255.224 {
       range dynamic-bootp 204.254.239.42 204.254.239.52;
       option broadcast-address 204.254.239.31;
       option routers snarg.fugue.com;
}

# Subnets can have no pooled ip addresses.
   subnet 10.10.10.0 netmask 255.255.255.0  {
   }
}

# IP subnets that are alone on their physical wire should be declared by
# themselves. The DHCP server still refers to them as shared networks in
# log messages, but this is simply an artifact of the underlying data
# structure.
```

```
#
# Note that options can be specified in the subnet declaration that
# supercede the global options specified earlier.

subnet 192.5.5.0 netmask 255.255.255.224 {
    range 192.5.5.26 192.5.5.30;
    option domain-name-servers bb.home.vix.com, gw.home.vix.com;
    option domain-name “vix.com”;
    option routers 192.5.5.1;
    option subnet-mask 255.255.255.224;
    option broadcast-address 192.5.5.31;
    default-lease-time 600;
    max-lease-time 7200;
}

# Hosts that require special configuration options can be listed in host
# statements. If no address is specified, the address will be allocated
# dynamically (if possible), but the host-specific information will still
# come from the host declaration.

host passacaglia {
    hardware ethernet 0:0:c0:5d:bd:95;
    filename “vmunix.passacaglia”;
    server-name “toccato.fugue.com”;
}

# Fixed IP addresses can also be specified for hosts. These addresses
# should not also be listed as being available for dynamic assignment.
# Hosts for which fixed IP addresses have been specified can boot using
# BOOTP or DHCP. Hosts for which no fixed address is specified can only be
# booted with DHCP, unless there is an address range on the subnet to
# which a BOOTP client is connected which has the dynamic-bootp flag set.
host fantasia {
    hardware ethernet 08:00:07:26:c0:a5;
    fixed-address fantasia.fugue.com;
}

# If a DHCP or BOOTP client is mobile and might be connected to a variety
# of networks, more than one fixed address for that host can be specified.
# Hosts can have fixed addresses on some networks, but receive dynamically
# allocated address on other subnets; in order to support this, a host
# declaration for that client must be given which does not have a fixed
# address. If a client should get different parameters depending on what
# subnet it boots on, host declarations for each such network should be
# given. Finally, if a domain name is given for a host’s fixed address and
# that domain name evaluates to more than one address, the address
# corresponding to the network to which the client is attached, if any,
# will be assigned.
host confusia {
    hardware ethernet 02:03:04:05:06:07;
```

```
    fixed-address confusia-1.fugue.com, confusia-2.fugue.com;
    filename “vmunix.confusia”;
    server-name “toccato.fugue.com”;
}

host confusia {
    hardware ethernet 02:03:04:05:06:07;
    fixed-address confusia-3.fugue.com;
    filename “vmunix.confusia”;
    server-name “snarg.fugue.com”;
}

host confusia {
    hardware ethernet 02:03:04:05:06:07;
    filename “vmunix.confusia”;
    server-name “bb.home.vix.com”;
}

# Some other examples
host host1 {
    option dhcp-client-identifier “host1”;
    fixed-address 10.10.11.101, 10.11.22.101;
}

# Do not allow this one to boot
host host2
    hardware ethernet aa:cc:04:00:33:11;
    deny booting;
}
```

DHCP Safe-failover Introduction

Since a DHCP server is responsible for the network’s IP management, it can also be a potential point of network failure if it becomes unavailable. Using multiple servers with non-overlapping IP address pools is one way to provide limited fault-tolerance. For example: imagine a network with two DHCP servers. Server A has an address range of 100 IP addresses. Server B has a range of 50 different addresses. Both servers have a non-overlapping range of addresses. When a node broadcasts for an address, both servers respond, each offering an address from its own distinct range. Upon receiving both offers, the node chooses one. Typically, the response that reaches the node first is selected. In this case, Server A’s. When Server B determines its offer is not needed, it returns the offered address to its own range, allowing it to be offered again.

If one of the servers is down, the other server continues to service the nodes. Now, instead of having two offers, each new node has only one offer, from the remaining server. Nodes that received their lease from the unavailable server attempt to reconnect with it. If the unavailable server does not respond in time, the nodes then attempt to get a new address from a new server. The other server can then offer an address from its own range. So, even though one server is down, the DHCP clients continue to function with the other server. Note that the two DHCP servers operate without any communications or data sharing between them. Each server works as a standalone server.

Since most nodes select the first offer received, having two servers could result in partial use of both IP address pools. Sometimes it is preferable to have a primary DHCP server with the bulk of the IP addresses while the secondary server has a smaller range of IP addresses.

Note! One way to accomplish the above mentioned configuration is to put the secondary server behind a router on a different subnet, while the primary server stays on the same segment as the nodes. This allows the primary server to respond more quickly than the secondary server.

Process Software takes the use of multiple servers to another level by offering DHCP Safe-failover. DHCP Safe-failover allows a secondary DHCP server to back up the primary DHCP server with the addition of taking into account network failure. This strategy insures that clients can reliably log into their corporate network and know they will be able to connect to corporate resources.

In Safe-failover mode both the primary and the backup DHCP servers share a common IP address lease pool. In the event the primary DHCP server fails the backup DHCP server automatically senses the primary server is not operating and automatically assumes complete DHCP operation. When the primary DHCP server becomes operational, it synchronizes with the backup DHCP server and resumes all the responsibilities of the primary DHCP server. All assignments performed by the backup DHCP server while acting as the primary server are transferred to the primary DHCP upon resumption of primary server responsibilities.

Safe-failover adds support for network, not server, failure. In the event the network fails, the secondary server believes the primary server is out of service and begins operation. The secondary server serves leases from a reserved pool shared by the safe-failover partner servers. This reserve pool can be adjusted by the MIS system administrator.

Configuring Safe-failover

To configure your DHCP servers to use Safe-failover, perform the following steps:

1	Choose one system to be the Primary and a second system to be the Secondary.
2	Determine the IP addresses of the Primary and Secondary systems. If a system has more than one IP address, choose one to use for DHCP Safe-failover messages.
3	On the Primary system, create the boot file at TCPWARE:DHCPD.BOOT with the keyword "primary", the primary and secondary IP addresses, and configuration ID. On the primary system, the configuration ID would normally be "dhcpd". See *Boot File for DHCP Safe-failover* for more information about boot files. Primary system boot file syntax: primary <primary-ip> <secondary-ip> "config-id";

4	Also on the Primary system, if there is no state file (see *State File for DHCP Safe-failover*), create one at TCPWARE:DHCPD.STATE containing the following line. Note that the name of this file must match the “config-id” parameter in the boot file. `primary-normal 0;`
5	On the Secondary system, create the boot file at TCPWARE:DHCPD.BOOT with the keyword "secondary", the secondary and the primary IP addresses, and configuration ID. On the secondary system, the configuration ID may be "dhcpd" or may be a name that refers to the primary. Either way, the names of the configuration, lease, and state files must match this name. Secondary system boot file syntax: `secondary <secondary-ip> <primary-ip> "config-id";`
6	Also on the Secondary system, if there is no state file (see *State File for DHCP Safe-failover*), create one at TCPWARE:DHCPD.STATE containing the following line. Note that the name of this file must match the “config-id” parameter in the boot file. `backup-normal 0;`
7	If you don't already have a configuration file, write a configuration file containing the subnets, shared networks, IP address ranges, hosts, etc, that reflect your network and the hosts you want the DHCP server to service. Include any DHCP Safe-failover parameters as needed (see *DHCP Safe-failover Configuration File Statements*).
8	Copy the configuration file to the TCPWARE directory on both the Primary and the Secondary systems. ***Note!*** Make sure the name of the configuration file matches the `config-id` parameter in the boot file for that system. Preferably, the configuration files on the Primary and the Secondary server systems should be the same. To help ensure that the configuration file is valid for both systems, make sure it contains a `subnet` statement for every subnet that either the Primary or the Secondary system has a network interface on.
9	Make sure that both the Primary and the Secondary systems have lease files in the TCPWARE directory with the name that matches `config-id`. If the lease file does not exist, create an empty one.
10	Run the DHCP server on both the Primary and the Secondary systems. The two servers will establish communications with each other and you're in business!

Boot File for Safe-failover

To use Safe-failover, create a boot file at TCPWARE:DHCPD.BOOT. The boot file is used to specify the following for Safe-failover operation:

- Server’s mode of operation
- Server’s own IP address
- Partner server’s IP address
- Configuration ID

The format of the boot file is:

```
mode [server-IP partner-IP] "config-id";
```

If the boot file is not present, the server operates with DHCP Safe-failover disabled. It uses tcpware:dhcpd.conf and tcpware:dhcpd.leases for its default configuration and lease files. In this state, the service parameters CONFIGFILE and LEASEFILE may be used to rename or move these files. The server does not use a state file to keep track and remember its state transitions when running in standalone mode (that is, with DHCP Safe-failover disabled). See *State File for DHCP Safe-failover* for a description of state files.

The following server modes are possible:

Table 2-5 DHCP Safe-failover Server Modes

In this mode	The server runs...
Primary	As the primary server, with DHCP Safe-failover functionality enabled. In this mode, the server tries to "shadow" each of its lease transactions to its configured secondary server.
Secondary	As the secondary or Backup server, with DHCP Safe-failover functionality enabled. The server receives lease transaction updates from its configured primary server, and maintains an up-to-the-minute hot backup of the lease database. If the primary server crashes, or is shut down, the secondary server takes over.
Standalone	With DHCP Safe-failover disabled.

The IP address following the server mode is the server address, the next IP address is the secondary or backup server IP address.

The configuration ID is the file name (without the file type) of the configuration, lease, and state files. For example, if the configuration ID is ALPHA, the DHCP server looks for a configuration file named ALPHA.CONF, a lease file named ALPHA.LEASES, and a state file named ALPHA.STATE.

```
primary 199.23.24.11 199.23.24.25 "ALPHA";
```

The example boot file designates the current server as the primary with its own IP address of 199.23.24.11 and the secondary server IP address of 199.23.24.25.

If the mode of operation is "standalone," the IP addresses are not needed. The format is `standalone "config-id";`

State File for DHCP Safe-failover

The state file is written by the DHCP server when it is running in DHCP Safe-failover mode. The state file saves server state changes so that a server can "remember" its last state if it crashes or is re-started. Alternately, you can use the state file to force the DHCP server to start up in a desired mode (operator override). This feature allows you to switch the server into partner-down mode without a lengthy time-out period, or to start up in recover mode (that is, to force the DHCP server to download the entire lease database from the partner).

The server appends a line to the state file every time its DHCP Safe-failover state changes. The last line in the file is the most current DHCP Safe-failover state.

The state file consists of one or more lines of the following format:

```
server_state transaction_count; [date-time;]
```

In the syntax, server_state is one of the following:

Table 2-6 DHCP Safe-failover Server States

- failover-disabled
- startup
- primary-normal
- backup-normal
- primary-comint
- backup-comint
- primary-partnerdown
- backup-partnerdown
- primary-conflict
- backup-conflict
- primary-recover
- backup-recover

Server-to-server messages are each assigned a monotonously increasing transaction number, which is recorded in the transaction_count field. This is an unsigned 32-bit number.

If the date-time stamp is present, the DHCP server assumes that the state was recorded by the server itself. In this case, the server, upon starting up, calculates the safest state based on the recorded state and the time elapsed. If the date-time stamp is not present, the DHCP server treats the entry as an operator-written override command and starts up in the specified state.

DHCP Safe-failover Lease File Statements

The statements shown in Table 2-7 have been added to the DHCP lease file for DHCP Safe-failover. These are in addition to the DHCP lease file statements listed in Table 2-4.

Table 2-7 DHCP Safe-failover Lease File Statements

Statement	Description
acked-sec-interval <u>seconds</u>;	Acknowledged secondary lease interval. For information, see the DHCP Failover internet draft.
acked-sec-interval-start *seconds;*	The time when the partner was notified of the lease.
active;	This IP address has a current lease.

Table 2-7 DHCP Safe-failover Lease File Statements (Continued)

Statement	Description
backup;	This IP address is reserved for use by the secondary (backup) server.
desired-interval <u>seconds;</u>	The length of the desired lease.
expired;	The lease for this IP address has expired.
free;	This IP address is available to be assigned.
released;	The lease for this IP address has been released by the client or by the operator.
reset;	The DHCP server had marked this IP address as abandoned. The operator changed its status to available.
safe-lease;	This is used in the Partner Down state to indicate that the IP address belongs to this server.
transaction-id <u>number</u>;	This is the transaction number that was assigned to this lease when it was queued or sent as an update to the partner server.
update-count <u>n</u>;	For each lease, the server which issues the lease sends an update to its partner server. This statement records the state of that update. 0 means no update is required 1 means that no update has been sent 2+ means 1 or more updates have been sent

DHCP Safe-failover Configuration File Statements

The statements shown in Table 2-8 have been added to the DHCP configuration file for DHCP Safe-failover. These are in addition to the DHCP configuration file statements listed in Table 2-1. All the parameters in Table 2-8 must be placed in the configuration file's global scope, with the exception of the backup-pool-size parameter. It may also be specified within a shared-network or subnet declaration.

Table 2-8 DHCP Safe-failover Configuration File Statements

Statement	Description
backup-ack-interval	The number of seconds used as the basis of the DHCP server's logarithmic back-off algorithm used for resending ACK messages to the secondary server. The default is 1 second. `backup-ack-interval seconds;`
backup-pool-size	This is the percentage of the IP address pool set aside for the Secondary server's emergency use. The DHCP server will reserve no more than 50% of the available addresses. The default is 10%. `backup-pool-size percent;`
com-int-timeout	The number of seconds the server should wait for an expected response from its partner before switching to communication interrupted mode. The default is 600 seconds (10 minutes). `com-int-timeout seconds;`
failover-port	The UDP port the Primary and Secondary servers should use for DHCP Safe-failover messages. The default is 647. `failover-port port;`
mclt	Maximum Client Lead Time: This is the length of lease in seconds to give out on a first time basis, or the client lead time for renewals. See the DHCP Failover internet draft for a more detailed explanation. The default is 3600 seconds (1 hour). `mclt seconds;`
safe-period-timeout	The number of seconds spent in communication interrupted state before automatic switch over to partner-down state. A value of 0 means no automatic switch over. The default is 0 seconds. `safe-period-timeout seconds;`
startup-delay	The number of seconds to wait during startup before the server moves from STARTUP state to the state specified in the state file. The default is 5 seconds. `startup-delay seconds;`

Transitioning to DHCP Safe-failover Partner-Down State

There are three ways that you can transition the DHCP server to Partner Down state, which indicates that its DHCP Safe-failover partner is down.

1	If the parameter safe-period-timeout is specified in the configuration file, the DHCP server transitions to Partner Down state automatically after it has been in Communication Interrupt state for the specified time.
2	You can put the DHCP server into Partner Down state using the following NETCU command: `$ `**`netcu set dhcp/partnerdown`**
3	You can edit the DHCP server state file and add a line to the end containing the Partner Down state and transaction number desired. Add either line: • primary-partnerdown transaction-number; • backup-partnerdown transaction-number; The next time the DHCP server is restarted, it will be in Partner Down state. You can restart the DHCP server by shutting down the server with the NETCU STOP/DHCP command. NETCP automatically restarts the DHCP server.

PART II Managing Networks

Chapter 3

Common Interfaces

Introduction

This chapter describes the following network interfaces:

- Ethernet
- Fiber Distributed Data Interface (FDDI)
- Token Ring
- LAN Emulation over ATM
- Classical IP over ATM
- HYPERchannel
- IP-over-DECnet
- proNET
- Compaq Wide Area Network (WAN) device drivers
- Pseudo devices

Alpha systems may not support all the interfaces supported by TCPware for OpenVMS software. See the *Installation & Configuration Guide*, Chapter 3, *Configuring the TCP/IP Core Environment*, Table 3-4.

Ethernet, FDDI, Token Ring, and ATM

TCPware for OpenVMS supports all Compaq Ethernet, FDDI, Token Ring, LAN Emulation over Asynchronous Transfer Mode (ATM), and Classical IP over ATM (CLIP) interfaces, so that you can send IP datagrams over these types of LANs.

These interfaces generally include the use of the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP), except CLIP, which does not support RARP.

For details on configuring LAN network devices and their addresses, see the *Installation & Configuration Guide*, Chapter 3, *Configuring the TCP/IP Core Environment*.

Address Resolution Protocol

ARP dynamically maps between internet and physical addresses. TCPware provides an ARP table of mappings that it keeps in cache. If a mapping is not in this cache when a datagram is transmitted, TCPware queues the datagram and broadcasts an ARP request over the network. When a host responds with an internet-to-physical address mapping, TCPware adds it to its cache and transmits the queued datagrams.

TCPware's implementation of ARP does not probe for impostors. However, it may generate the following OPCOM message, which includes the physical address of the system with a duplicate internet address:

```
Duplicate IP address! Sent from physical address...
```

In the rare case where the remote system does not support ARP, you can use the ADD ARP and REMOVE ARP commands in TCPware's Network Control Utility (NETCU) to add or remove entries for the system. You can also find specific ARP entries using FIND ARP, and show the entire ARP table using the SHOW ARP command.

You can also set various ARP parameters using /ARP_*x* qualifiers with the SET INTERFACE command. The ARP qualifiers control when to check the age of an ARP entry, how long to keep it in cache, how long to wait for an unresolved entry to be removed from cache, and the maximum size of the cache. TCPware provides defaults for these parameters, so you do not normally need to use these commands. For example, it normally removes entries from its ARP table if it does not receive a packet for an entry within 10 minutes, or if the table is more than 512 entries long.

Reverse Address Resolution Protocol

The Reverse Address Resolution Protocol (RARP) enables a diskless client to find its IP address at startup from a RARP server. The diskless client broadcasts a request that contains its physical hardware address. The server maps the hardware address to the IP address corresponding to the physical address of the client. The TCPware system only responds to RARP requests for permanent address entries in its ARP cache. (Note that BOOTP provides the same type of services.)

RARP support is enabled by default for all Ethernet, FDDI, and Token Ring interfaces. RARP support is disabled for LAN Emulation over Asynchronous Transfer Mode (ATM) and Classical IP over ATM (CLIP-*n*) lines. You can explicitly disable RARP support using the NETCU START/IP command with the /FLAGS=NORARP qualifier.

Ethernet Trailer Packets

TCPware can operate with trailer packets enabled or disabled. Trailer packets have some of the packet headers at the end of the packet rather than at the beginning. TCPware disables trailer packet support by default on Ethernet lines. However, some implementations use trailer packets, such as those running under UNIX. TCPware can receive and process trailer packets, but it will never transmit them.

To disable trailer packets on UNIX systems, use the `ifconfig` command with the `-trailers` option. To disable trailer packets on OpenVMS systems, use the NETCU START/IP /FLAGS=NOTRAILERS qualifier.

Qualifiers with LAN Device Lines

The START/IP command supports a number of qualifiers that you can use with Ethernet, FDDI, Token Ring, LAN Emulation over Asynchronous Transfer Mode (ATM), and Classical IP over ATM lines. See Table 2-11 and Table 2-13 in the *NETCU Command Reference*.

VMS Communications Interfaces Support

TCPware supports VMS Communications Interfaces (VCIs). VCI is a high speed interface to the LAN drivers. If you want to disable VCI support for some reason, use the /FLAGS=NOVCI qualifier to the NETCU START/IP command.

Limiting Receive Packet Rate

TCPware supports placing a limit on the number of receive packets it processes per second. If a limit is set on an interface and that limit is exceeded, TCPware may issue the following OPCOM message:

```
Warning - maximum receive packet rate exceeded on line line-id (rate
packets/second).
```

This indicates that the interface specified by *line-id* received more packets than were allowed. This may indicate that either the receive packet rate limit is too low or that a flood of packets arrived at the system and a network problem exists that should be corrected. If the limit is too low, raise it using the NETCU SET INTERFACE /RECEIVE_LIMIT command. If a network problem exists, investigate it and correct it.

HYPERchannel

TCPware supports any HYPERchannel interface supported by Network System Corporation's H269 device driver, including the UNIBUS, QBUS, MASSBUS, and BIBUS interfaces.

The HYPERchannel interface support includes the use of the Address Resolution Protocol (ARP). Use ARP to automatically map an internet address to a physical address.

Address Format

When starting a HYPERchannel line, you must specify line-specific-information. This is the local HYPERchannel address in 32-bit address. The format is ***aa-bb-cc-dd***, where *aa*, *bb*, *cc*, and *dd* are hexadecimal values representing each byte of the address:

- *aa* is the global network address domain (if none, specify `00`)
- *bb* is the global network address network (if none, specify `00`)
- *cc* is the physical unit
- *dd* is the logical unit

If you are using 16-bit addresses, specify the address as `00-00-`*`cc-dd`*.

The H269 driver's IO$_ATTACH function uses the *cc-dd* portion of the local HYPERchannel address as the path address. You should always specify the local HYPERchannel address parameter as the 32-bit HYPERchannel address.

Address Mapping

TCPware needs to map the 32-bit internet addresses into 32-bit HYPERchannel addresses. These two address families are not related. That is, there is no mathematical formula that you can use to convert from one address family to the other. Instead, you must either configure an ARP server or pre-load the address resolution table with the mappings.

Note! You must properly configure the server with address mappings for all systems on the HYPERchannel network.

To pre-load the address resolution table, use NETCU ADD ARP commands (see the *NETCU Command Reference*). The ADD ARP command requires that you specify a 48-bit value (in hexadecimal format). The syntax is:

`aa-bb-cc-dd-ee-ff`

where the additional *ee* is the HYPERchannel trunks-to-try mask (typically `FF`) and *ff* is the HYPERchannel flags mask (typically 00). For ease of use, the ADD ARP command allows you to specify most addresses using *`aa-bb-cc-dd`*-00-00. It supplies the proper trunks-to-try mask of `FF`. See the HYPERchannel documentation for information on other values for the trunks-to-try and flags fields.

Note that if you load the address resolution table through the ADD ARP command, specify /PERMANENT. Otherwise, TCPware removes the added entries after a short time.

Qualifiers with HYPERchannel Lines

The START/IP command supports a number of qualifiers that you can use with HYPERchannel lines. See Table 2-11 in the *NETCU Command Reference*.

IP-over-DECnet

TCPware for OpenVMS provides support for DECnet interface implementations so that you can send IP datagrams over DECnet links. This lets you connect separate TCP/IP LANs over DECnet WAN links.

Configuring DECnet Lines

To configure a DECnet line:

1	Enter the DECnet line identification, internet address, and host name for the local internet address in the response to the applicable prompts in the CNFNET network configuration utility. Note that you need a different IP address for the DECnet line.
2	Enter the appropriate line-specific information.

Line-Specific Information

The START/IP command *line-specific-information* parameter provides the required DECnet link information. Enter the *line-specific-information* in the following format:

***node-name*::"TASK=*object-name*"**

Parameter	Identifies the...
node-name	listener node when the master mode issues it. It identifies the master node when the listener mode issues it.
object-name	object used on the listener node. Both the master and listener nodes must specify the same *object-name*.

An IP-over-DECnet line has a master node at one end and a listener node at the other end.

Sample Configuration

Example 3-1 shows selections from a sample IP-over-DECnet configuration in TCPware's CNFNET.

Example 3-1 Sample Configuration in CNFNET

```
Line Id       Network Device
QNA-n        for Compaq's DELQA, DESQA, or DEQNA (XQDRIVER)
UNA-n        for Compaq's DELUA or DEUNA (XEDRIVER)
.
.
.
Enter the line identifications [LPB-0,ISA-0]: LPB-0,ISA-0,DECNET-0
.
.
.
What is the local host's INTERNET ADDRESS for line ISA-0 [172.16.1.1]:
What is the NAME for line ISA-0 [BVA2]: BVA2
What is the SUBNET MASK for line SVA-0 [255.255.0.0]:
Do you want to enable TRAILER packet support for line ISA-0 [NO]:
Do you want to enable RARP (Reverse ARP) support for line ISA-0 [YES]:

What is the local host's INTERNET ADDRESS for line DECNET-0: 172.16.1.2
What is the NAME for line DECNET-0: BVA2
What is the SUBNET MASK for line DECNET-0 [255.255.0.0]:
What is the DECnet link information for line DECNET-0:
ONA1::"TASK=OZONE"
Is this the LISTENER end of the DECnet link for line DECNET-0 [NO]:

The network devices are configured as follows:

    Line     Address       Name         Options
    ----     -------       ----         -------
```

```
        LPB-0    127.0.0.1     LOOPBACK
        SVA-0    172.16.1.1    BVA2          /MASK=255.255.0.0
                                             /FLAGS=(NOTRAILERS)
        DECNET-0 172.16.1.2    ONA1          /MASK=255.255.0.0
        ONA1::"task=OZONE"/FLAGS=LISTENER
    .
    .
    .
```

Qualifiers with DECnet Lines

The START/IP command supports a number of qualifiers that you can use with DECnet lines.

See Table 2-11 in the *NETCU Command Reference*.

proNET-10/80

The proNET-10 and proNET-80 token ring controllers produced by Proteon, Inc. form a link between the hardware devices, the token ring, and TCPware.

Configuring proNET Lines

When you configure proNET lines, the D component of the internet address (using the standard A.B.C.D convention) must match the node address of the proNET controller.

Qualifiers with proNET Lines

The START/IP command supports a qualifier that you can use with proNET lines.

See Table 2-11 in the *NETCU Command Reference*.

Compaq Wide Area Network (WAN) Device Drivers

The Compaq WAN Device Drivers are synchronous interfaces that form a link between the hardware devices and TCPware. TCPware for OpenVMS supports the DSV11, DSB32, and DST32 Compaq WAN interfaces.

Line-Specific Information

The NETCU START/IP command's *line-specific-information* parameter provides the required DECnet link information. The *line-specific-information* is a quoted string of the line configuration options shown in Table 4-1.

For details on these parameters, such as the possible values for the line speed, CRC, and so on, see Compaq's *VAX Wide Area Network Device Drivers Programmer's Guide*.

An example of *line-specific-information* is:

```
"PROTOCOL DDCMP POINT CLOCK INTERNAL LINE SPEED 64000"
```

When specifying *line-specific-information* for Compaq WAN Device Drivers lines, be aware of the following:

- You must enclose *line-specific-information* in quotes for these lines.
- You can use keyword abbreviations.

Qualifiers with VAX WAN Device Driver Lines

The START/IP command supports several qualifiers you can use with Compaq WAN Device Driver lines. See Table 2-11 in the *NETCU Command Reference*.

Table 3-2 WAN Parameters

Parameter	**Takes...**	**Description**
PROTOCOL	DDCMP POINT LAPBE LAPB SDLC	Line protocol used
DUPLEX	HALF FULL	Line operation used
CLOCK	INTERNAL EXTERNAL	Line clocking used
CRC	type	Type of CRC used (*not recommended*)
LINE SPEED	baud	Line speed (only useful with `CLOCK INTERNAL`)
RECEIVE BUFFERS	number	Number of receive buffers
RETRANSMIT TIMER	time	Retransmission time (for `PROTOCOL DDCMP POINT` only)

Pseudo Devices

Pseudo devices are a way to configure a physical device to have multiple Internet addresses. Pseudo devices are typically used when a system is connected to a network that needs extra network numbers assigned to it. Pseudo devices can also be used in place of secondary addresses (for example, when a system has multiple addresses on the same network).

When starting a pseudo device, you specify the local Internet address, network mask, and the physical device to which the pseudo device is connected.

Adding a Pseudo Device

CNFNET has been updated to allow for configuring pseudo devices.

The TCPware line-id for a pseudo device is PSD-*n*, and *n* is 0 to 255.

To configure one or more pseudo devices via CNFNET, include the line-id or line-ids for the pseudo devices when prompted to enter the line identifications for all the network devices. Be sure to enter the pseudo device line-id *after* the physical device line-id. CNFNET prompts you for the standard information (Internet address, host name, and subnet mask) and for the physical device line-id for the pseudo device.

You can also start pseudo devices by using the NETCU START/IP command. See START/IP in Chapter 2 of the *NETCU Command Reference*.

Characteristics of Pseudo Devices

Pseudo devices are interchangeable and usable just like physical devices.

However, there are a few special characteristics that are important to point out:

- Multicast joins/leaves are redirected to the physical device.
- NETCU SHOW NETWORK shows no transmit/receive counts for pseudo devices. The physical device reflects the transmit/receive activity.
- Pseudo devices are automatically removed whenever the physical device is removed (such as by a NETCU STOP/IP line-id command).
- Starting a pseudo device on a pseudo device (by specifying a pseudo device line-id as the *Real-Line-ID* for a NETCU START/IP command) is allowed; however, the underlying physical device is actually used.
- Packet filtering is not available for pseudo devices as these devices never receive any packets (the physical device does). Therefore, you must do all packet filtering on the physical device and must take this into consideration when creating the packet filter list. Attempting to issue a NETCU SET FILTER or NETCU SHOW FILTER command on a pseudo device returns an error message.
- Once a pseudo device is started, there is no indication available as to the physical device with which it is associated.
- The line-id value for pseudo devices is 00*nn*0042 (hex), where *n* is the unit number (PSD-*n)*.
- For proper operation of pseudo devices (and TCPware in general), the LPB-0 (loopback) device must exist. The LPB-0 device is technically not optional.
- Pseudo devices can not be started on unnumbered interfaces.

When to Use Pseudo Devices, Secondary Addresses, and Interface Routes

TCPware continues to support secondary addresses (NETCU ADD SECONDARY) and interface routes (NETCU ADD ROUTE) in addition to the new pseudo devices. Some recommendations as to which method to use and the conditions under which to use them are described next.

- If a TCPware system is connected to a network via a single interface that has multiple networks numbers assigned to it:
  - Use a pseudo device for each network number on which the TCPware system has an Internet address (other than the one that is used to start the physical device).
  - Use an interface route for each network number on which the TCPware system does not have an Internet address. For an interface route, specify the line-id of the physical device

in place of the gateway address parameter.

- If a TCPware system has multiple addresses on a single network number:
  - Use either pseudo devices or secondary addresses for the additional addresses. Using a pseudo device has some advantages and is recommended (especially if a DNS server is running on the system).
- To use the cluster alias failover support:
  - The secondary address feature must be used.

Note! If your site is using secondary addresses you might want to consider whether switching to pseudo devices makes sense for these addresses.

Chapter 4

Serial Link Interfaces: PPP and SLIP

Introduction

This chapter describes the serial link interfaces available with TCPware. There are two types of serial link interfaces:

- Point-to-Point Protocol (PPP)
- Serial Line IP (SLIP)

Point-to-Point Protocol Interface

TCPware supports the Point-to-Point Protocol (PPP) so that you can send IP datagrams over serial links, including LAT or modem connections.

PPP is an enhancement to the nonstandard Serial Line IP (SLIP) interface (see *Serial Line IP Interface*). PPP provides self-contained error detection and automatically negotiated header compression. It also provides authentication through the Password Authentication Protocol (PAP) or Challenge Handshake Authentication Protocol (CHAP), which you can set using PPPD command options.

You configure PPP on the TCPware host using the PPPD command and its options at the DCL prompt or aggregated in an options file. You do not need to configure PPP using the configuration procedure (CNFNET) or Network Control Utility (NETCU) commands.

Implementation

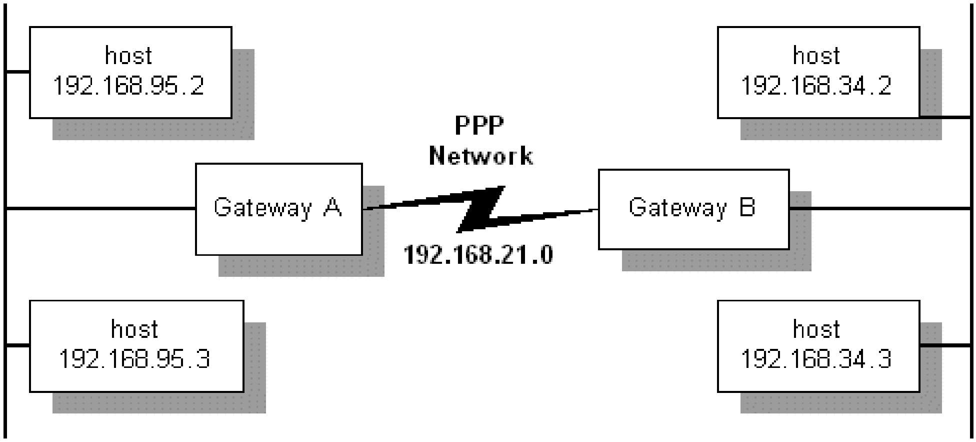
PPP is a standard (as Internet STD 51, or RFC 1661) for transporting multiprotocol datagrams over serial point-to-point links. PPP is composed of three parts:

- A method to encapsulate multiprotocol datagrams over serial links.
- An extensible Link Control Protocol (LCP) to establish, configure, and test the data link connection.
- A family of Network Control Protocols (NCPs, such as the IP Control Program and the

authentication protocols) to establish and configure the different network layer protocols.

You need TCP/IP and application software at each end of the PPP link. One end can be TCPware, while the other end (known as the *peer*) can be any PPP client implementation. Figure 4-1 shows a typical PPP network.

Figure 4-1   Sample Internet with PPP Link



Because PPP lines are point-to-point connections, with TCPware you have the option to configure PPP lines as network lines or as unnumbered interface lines. The advantage to unnumbered interface lines is that you do not need to assign them IP addresses.

TCPware supports both dedicated and dialup PPP lines. PPP is common with line speeds from 14.4 to 28.8 kilobits per second (Kb/s).

PPP is implemented as a single program, PPPD, the PPP daemon, which runs as a foreign command. The daemon is a process that:

- Runs between the terminal driver and IP driver.
- Negotiates PPP line configuration with a peer PPP node.
- Establishes the PPP connection between the specified serial line and TCPware.

There is one PPPD process for each physical connection.

You control PPP through PPPD command line options, which you can add to an options file (TCPWARE:PPPOPTIONS.DAT).

Before Configuring PPP Lines

Before you begin configuring PPP lines:

- You must set the TTY_ALTYPAHD system parameter larger than its default value. This avoids losing characters. The higher the line speed, the higher you should set this parameter. For most applications, 1024 is appropriate.

TTY_ALTYPAHD is not a dynamic parameter. If you use SYSGEN to change it, you must reboot the system for the change to take effect.

- The MAXBUF system parameter must be at least twice the maximum transmission unit (MTU) of the PPP line plus 134. The default MTU for PPP lines is 1500 bytes; therefore, MAXBUF must be at least 3134. Increase MAXBUF if necessary. MAXBUF is dynamic. If you use SYSGEN to change it, you do not need to reboot the system for the change in value to take effect.
- Determine the speed of your serial line or modem. You may need to specify this line speed as a PPPD command parameter.
- Make sure that the terminal device you specify is allocated to your current process.

PPPD Command

You configure PPP lines on your local system using the PPPD command at the DCL prompt and specifying the appropriate options. You start PPPD as a foreign command as follows:

```
$ PPPD :== $TCPWARE:PPPD.EXE
$ PPPD options
```

When you start PPPD, it either begins sending PPP packets to the specified terminal device to start negotiating, or waits for PPP packets to arrive, depending on the *options* specified on the command line. When negotiation is complete and IP is up, you can tell if the PPP device is configured by using the NETCU command SHOW NETWORK:

```
$ NETCU SHOW NETWORK

TCPware(R) for OpenVMS Internet Network Information:
Line   Local Address  Subnet Mask    MTU    Xmits   Errs  Recvs Errs  RBU
----   -------------  -----------    ---    -----   ----  ----- ----  ---
PPP-0  192.168.142.57 255.255.255.0  1500   1       0     1     0     0
LPB-0  127.0.0.1      255.0.0.0      64512  74      0     74    0     0
...
```

The PPP line shows up as PPP-*n*, the *n* starting with 0 and incrementing for each new line.

A PPP session is terminated in one of the following ways:

- If you enter the NETCU STOP/IP command for the PPP line
- The peer terminates the PPP session
- The serial line is hung up (for modem lines)
- The PPPD process is stopped

The PPPD process either ends or listens on the line for more incoming PPP packets, depending on the command line option used. PPP also ends when it cannot agree on the option negotiation during startup. An example is when the peer requires CHAP authentication and the local host only has PAP authentication configured.

By default, PPPD creates a detached process to which it hands over the terminal device. To execute PPPD in foreground mode, use the -DETACH (or -NODETACH) option with the PPPD command. (Note that special privileges apply to a detached process. See the -DETACH option in the PPPD command reference for details.) The name of the detached process is PPP_*terminal-device-name*.

The TCPWARE:PPPOPTIONS.DAT file can contain any option you can specify on the PPPD command line. Options in the file have precedence over the options on the command line. The options file can contain any of the PPPD options, separated with spaces or tabs. You can specify options in multiple lines, as in the following example:

```
NETMASK 255.255.252.0 ASYNCMAP 0
NAME SIRIUS.PROCESS.COM
AUTH +CHAP
```

You can specify command files in four different PPPD options, as described in Table 4-1.

Table 4-1   PPPD Options with Command File Parameters

Option	**Description**
CONNECT *file*	Sets the terminal device through the specified command file
DISCONNECT *file*	Resets the terminal device through the specified command file
IPUP *file*	The specified file executes when IP is started over PPP
IPDOWN *file*	The specified file executes when IP over PPP is shut down

The command file associated with the CONNECT and DISCONNECT options must return an exit status to PPPD with the EXIT command. PPPD waits for the script to finish and terminates if the return status is not successful (the status code is an odd number). The command files are provided with the device name as the P1 parameter. You can use a dialup scripting program such as KERMIT.

The IPUP and IPDOWN options are executed asynchronously so that PPPD does not wait for their completion. They are provided with the following command parameters:

Parameter	**Description**
P1	Interface name (such as PPP-0)
P2	Terminal device (such as _TTA3:)
P3	Local IP address
P4	Remote IP address
P5	(Optional) String specified with the IPPARAM option

Configuring PPP Links

The PPPD command line parameter and options shown in Table 4-2 control the PPP link configuration and specify the basic characteristics of the PPP link.

Table 4-2 PPP Link Basic Configuration Options

Option	Description
device-name	Terminal device, such as TTA3
ASYNCMAP *map*	Bit map of characters to escape
CRTSCTS	Uses hardware flow control
ESCAPE *xx,yy,...*	Escape character definitions
MRU *n*	Maximum Receive Unit (MRU)

See *Command Reference* for details on each option.

By default, PPPD starts sending out configuration requests to the peer to establish a connection, and terminates when the connection shuts down, negotiation fails, or the peer does not respond within a set timeout period. You can change this course of action using the options shown in Table 4-3.

Table 4-3 Link Customization Options

Option	Description
PASSIVE	Initiates negotiation but waits passively for the peer to respond if that fails
SILENT	Passively waits until the PPP packet arrives
PERSIST	After the PPP connection is terminated, waits for a new connection without exiting (in the case of a dial-out, reestablishes the connection by redialing)

Authentication

PPPD provides sufficient access control. You can provide legitimate users PPP access to a server machine, without fear of compromising the security of the server or the network it is on. This access control is available as a combination of the following:

- The TCPWARE:PPPOPTIONS.DAT file, where you can place options to require authentication when running PPPD.
- Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP) secrets files (see *Using the Challenge Handshake Authentication Protocol (CHAP)*), where you can restrict the set of IP addresses for individual users

PPPD's default action is to agree to authenticate if requested, and not to require authentication from the peer. However, PPPD does not agree to authenticate itself with a particular protocol if it has no secrets it can use to do so.

You can change this behavior with the command line options shown in Table 4-4.

Table 4-4 Link Authentication Options

Option	Description
AUTH	Peer authenticates (any authentication)
+CHAP	Peer authenticates with CHAP
-CHAP	Do not agree to authenticate with CHAP
+PAP	Peer authenticates with PAP
-PAP	Do not agree to authenticate with PAP

Using the Password Authentication Protocol

The Password Authentication Protocol (PAP) provides a simple method for the peer to establish its identity. PAP uses a two-way handshake with a simple name and password combination. This handshake occurs only on establishing the initial PPP link.

PAP is not a strong authentication method. Passwords go over the circuit as clear text, and there is no protection from playback or repeated trial and error attacks.

Note! You can optionally authenticate using the user/password combination in the OpenVMS UAF file by option `"login"`.

Using the Challenge Handshake Authentication Protocol

The Challenge Handshake Authentication Protocol (CHAP) is a stronger method than PAP of authenticating the PPP link, and is the preferred method. A CHAP secret (password) is encrypted, and you can repeat authentication periodically during the session using different challenge values.

Authentication Files

Authentication information is stored in either the TCPWARE:PPPCHAP.DAT file for CHAP or the TCPWARE:PPPPAP.DAT file for PAP authentication. The contents of these files are used both for authenticating incoming peer hosts and authenticating the local host to remote peers. You can use the TCPWARE:PPPSECRET.TEMPLATE file as a template for both, since the format for both is identical. The following example shows a PPPCHAP.DAT entry that authenticates the local host to a peer:

```
# TCPWARE:PPPCHAP.DAT
#
# local/user name   server name   secret/password   [address restriction]
```

```
# --------------   -----------   --------------   --------------------
  skat             lear          SecretString     198.168.142.57
```

Both files are multicolumned text files. Comment lines in PPPCHAP.DAT start with the number sign (#). Each line consists of three fields with additional optional fields, as follows:

field1 field2 field3 optional-fields

White space separates the fields. You can use the asterisk (*) wildcard in the first and second fields. Optional fields can contain lists of authorized peer IP addresses. If the optional field is omitted, any IP address is allowed.

The fields have different meanings for PAP and CHAP authentication, and its direction, as given in Table 4-5.

Table 4-5 PAP and CHAP Authentication File Fields

PAP authentication of incoming peer:			
field1 local-username	field2 peer-hostname	field3 user-password	optional
PAP authentication of local host to peer:			
field1 peer-username	field2 peer-hostname	field3 user-password	optional
CHAP authentication of incoming peer:			
field1 peer-hostname	field2 local-hostname	field3 CHAP-secret-string	optional
CHAP authentication of local host to peer:			
field1 local-hostname	field2 peer-hostname	field3 CHAP-secret-string	optional

Modifying Authentication Names

You can use several PPPD command line options to modify names used in authentication. These options are shown in Table 4-6.

Table 4-6 Authentication Name Modification Options

Option	**Description**
DOMAIN *d*	Appends the domain name *d* to the local hostname

Table 4-6 Authentication Name Modification Options (Continued)

NAME *n*	Sets the local hostname to *n*
REMOTENAME *n*	Sets the assumed remote hostname to *n*
USEHOSTNAME	Uses the system-specified local host name
USER *u*	Sets the username to *u* for PAP

IP Addresses

The parameter and options shown in Table 4-7 are related to configuring IP addresses.

Table 4-7 IP Address Configuration Options

Parameter or option	**Description**
local-IP-address:*remote-IP-address*	Local and remote IP addresses; either can be omitted
-IP	Disables IP address negotiation
NETMASK *n*	Sets the interface mask to *n*
NOIPDEFAULT	Disables use of the local IP address as the default

It is usually not necessary to specify the IP addresses. By default, each peer uses its default IP address if it knows it.

Incoming Dialup Lines

Perform these steps to set up an incoming dialup PPP line:

1	Create a login account for the PPP site using the OpenVMS AUTHORIZE utility. This should be a captive account and must have OPER privileges.
2	Create a LOGIN.COM file for this account. The TCPWARE:PPPLOGIN.TEMPLATE file is available. Figure 4-2 shows an unnumbered interface example.

Figure 4-2 Sample LOGIN.COM File

```
$ ON WARNING THEN LOGOUT
$ IF (F$TRNLNM("TCPWARE_NETCP_MBX") .EQS. "") THEN GOTO NOTCPWARE
$ PPPD :== $TCPWARE:PPPD
$ TT = F$TRNLNM("TT")
$ WRITE SYS$OUTPUT "Starting PPP..."
$ DEFINE/USER SYS$ERROR NLA0:
$ DEFINE/USER SYS$OUTPUT NLA0:
```

```
$ PPPD PROXYARP :192.168.95.12 'TT' -DETACH
$ WRITE SYS$OUTPUT "Shutting down PPP..."
$ EXIT
$ NOTCPWARE:
$ WRITE SYS$OUTPUT "%PPP-F-NOTACT, TCPware not active"
$ LOGOUT
```

Once you set up the account and login file, the remote site dials the OpenVMS system and logs in as the PPP user to establish the connection. The login command file automatically configures the PPP line.

Create a separate account and LOGIN.COM file for each remote PPP site.

Note! Unlike TCPware's SLIP implementation, this account can remain logged in during the PPP session. You can use ordinary OpenVMS user accounting to charge the user for the PPP connection. If you prefer to free up login sessions, you can remove the -DETACH option.

Routing

The options shown in Table 4-8 are related to configuring routes.

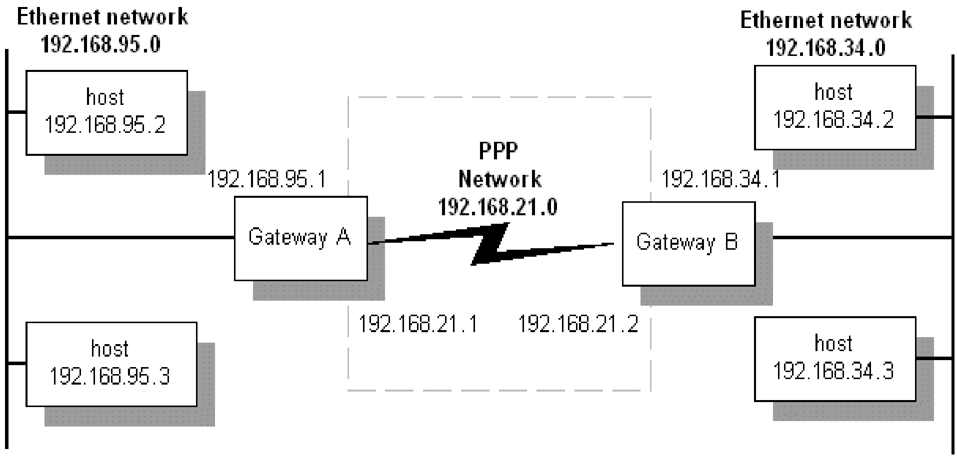
Table 4-8 Routing Configuration Options

Option	Description
DEFAULTROUTE	Uses the remote host as the default gateway
-IP	Disables IP address negotiation
NETMASK *n*	Sets the interface mask to *n*
PROXYARP	Starts the line as an unnumbered interface and enables proxy ARP
-PROXYARP	Disables proxy ARP

Traditional Numbered Interfaces

Figure 4-3 shows a sample internet consisting of three networks: Ethernet network 192.168.95.0, PPP network 192.168.21.0, and Ethernet network 192.168.34.0.

Figure 4-3   Sample Internet with PPP Link



Each gateway has an internet address for each network to which it connects. In this example, PPP network 192.168.21.0 is set up so that networks 192.168.95.0 and 192.168.34.0 can communicate.

Initiate the PPP link from local host 192.168.95.1 to peer 192.168.34.1 as follows:

```
$ PPPD 192.168.21.1:192.168.21.2 NETMASK 255.255.255.0 TTA2
```

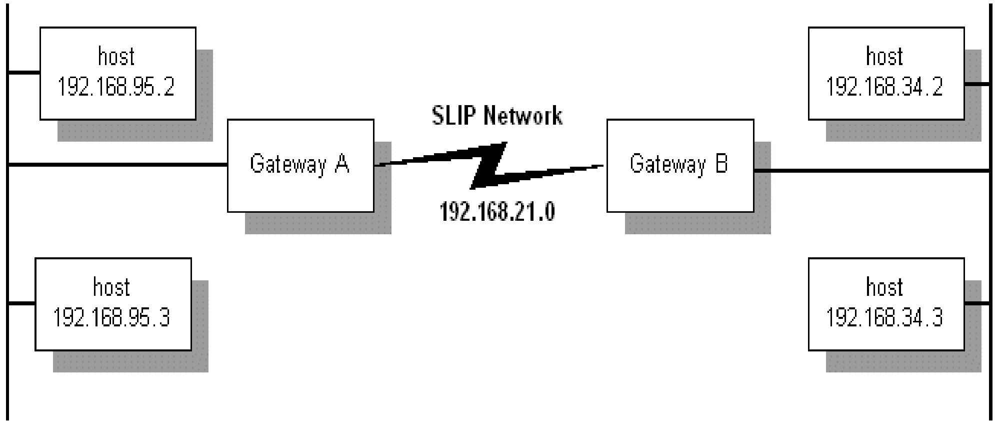
Perform a similar command on local host 192.168.34.1. You must also set up routing through the established connection. To do so in this setup, create a command file with the following contents for Gateway B, and use the PPPD IPUP option to specify the command file:

```
$ RUN TCPWARE:NETCU ADD ROUTE 192.168.34.0 192.168.21.1 /NETWORK /GATEWAY
```

Unnumbered Interfaces

In the case of a single host connection over a PPP line to a network, you may not be able to dedicate a separate network number to the connection. This requires you to use an unnumbered interface. Figure 4-4 shows such a scenario.

Figure 4-4 Sample PPP Connection with an Unnumbered Interface



Here is how you would set up the connection on host 192.168.34.1:

```
$ PPPD 192.168.34.1:192.168.34.4 NETMASK 255.255.255.0 PROXYARP TTA2
```

The PROXYARP option lets host 192.168.34.1 respond to Address Resolution Protocol (ARP) requests for the remote host's address. In this way, other hosts on the 192.168.34.0 network can send any packets addressed to 192.168.34.2. This is known as "proxy ARP" and keeps you from having to add the host routes on all other hosts in network 192.168.34.0.

The PROXYARP option does the following:

- Starts the PPP interface as unnumbered
- Enables proxy ARP for the remote node
- Adds a host route to the remote node through the PPP interface

TCP/IP Header Compression

PPPD also provides the option to compress TCP/IP headers using the Van Jacobson (VJ) header compression algorithm. Compression is enabled unless you for some reason want to disable it. The options shown in Table 4-9 are related to TCP/IP header compression.

Table 4-9 TCP/IP Header Compression Options

Option	Description
-VJ	Disables VJ compression
-VJCCOMP	Disables VJ Connection ID compression
VJ-MAX-SLOTS *n*	Sets the number of VJ connection slots to *n*

Command Reference

The following pages include the command reference for the PPPD command. The *options* include parameters and options (some with arguments). Parameters and options are listed alphabetically.

Frequently used parameters and options include:

device-name	*local-IP-address*:*remote-IP-address*			ASYNCMAP *map*
AUTH	CRTSCTS		DEFAULTROUTE	ESCAPE *xx,yy,...*
MRU *u*	MTU *u*		NETMASK *n*	PASSIVE
SILENT				

PPPD

Provides the basic Link Control Protocol (LCP), authentication support, and a Network Control Protocol (NCP) for establishing and configuring the IP Control Protocol (IPCP). Parameters and options are listed alphabetically for reference.

Requires OPER and PHY_IO privileges. Requires READALL privilege if secret files are used. Requires TMPMBX, DETACH, and SHARE privileges for running in detached mode.

Format

PPPD *[parameters* | *options]*

Parameters

device-name

Communicates over the named device. If no device name is given, or the name of the controlling terminal is given, PPPD uses the controlling terminal.

local-IP-address:remote-IP-address

Sets the local or remote interface IP addresses. Either one may be omitted. The IP addresses can be specified with a hostname or in decimal dot notation (such as 150.234.56.78). The default local address is the (first) IP address of the system (unless the NOIPDEFAULT option is given). The remote address is obtained from the peer if not specified in any option. Thus, in simple cases, this option is not required. If a local and/or remote IP address is specified with this option, PPPD does not accept a different value from the peer in the IPCP negotiation, unless the IPCP-ACCEPT-LOCAL or IPCP-ACCEPT-REMOTE options are given, respectively.

Options

-AC

Disables address/control compression negotiation (default).

-ALL

Disables requesting or allowing negotiation of any options for LCP and IPCP (uses the default values).

-AM

Disables ASYNCMAP negotiation (uses the default ASYNCMAP, which is to escape all control characters).

ASYNCMAP *map*
-AS *map*

Sets the async character map to *map*, which describes which control characters cannot be successfully received over the serial line. The peer is requested to send these characters as a two-byte escape sequence. The argument is a 32-bit hex number with each bit representing a character to escape. Bit 0 (00000001) represents the character `0x00`; bit 31 (80000000) represents the character `0x1f` (Ctrl/_). If multiple ASYNCMAP options are given, the values are ORed together. If no ASYNCMAP option is given, no async character map will be negotiated for the receive direction; the peer should then escape all control characters.

AUTH

Requires the peer to authenticate itself before allowing network packets to be sent or received.

+CHAP

Requires the peer to authenticate itself using CHAP (Challenge Handshake Authentication Protocol) authentication.

-CHAP

Disables authentication using CHAP.

CHAP-INTERVAL *n*

Rechallenges the peer every *n* seconds using CHAP.

CHAP-MAX-CHALLENGE *n*

Sets the maximum number of CHAP challenge transmissions to *n* (default 10).

CHAP-RESTART *n*

Sets the CHAP restart interval (retransmission timeout for challenges) to *n* seconds (default 3).

CONNECT *command-file*

Uses a DCL command file to set up the serial line. If used with -D, debug output is logged in the *command-file*.LOG file.

CRTSCTS

Uses hardware flow control (RTS/CTS) to control the flow of data on the serial port. If you use neither CRTSCTS nor -CRTSCTS, the hardware flow control setting for the serial port is unchanged.

-CRTSCTS

Disables hardware flow control (RTS/CTS) on the serial port. If you use neither CRTSCTS nor -CRTSCTS, the hardware flow control setting for the serial port is unchanged.

DEBUG
-D

Enables debugging.

DEFAULTROUTE

Adds a default route to the system routing tables, using the peer as the gateway, when IPCP negotiation is successfully completed. This entry is removed when the PPP connection breaks.

-DEFAULTROUTE

Disables the defaultroute option. If you want to prevent users from creating default routes with PPPD, place this option in the TCPWARE:PPPDOPTIONS.DAT file.

-DETACH
-NODETACH

Executes PPPD in foreground mode. Normally, PPPD creates a detached process to which it hands over the terminal device. To start in detached mode, PPPD requires DETACH privilege, along with TMPMBX and SHARE privileges if you are using your login device as a PPP device. -DETACH and -NODETACH are identical.

DISCONNECT *command-file*

Runs the DCL command file after PPPD terminates the link. This command file could issue commands to the modem to hang up, if hardware modem control signals were not available. If used with -D, debug output is logged in the *command-file*.LOG file.

DNS *address*

Identifies the primary Domain Name System (DNS) name server. If omitted, PPPD uses the first name server specified by the TCPWARE_NAMESERVERS logical, set up through Domain Name Services configuration.

DOMAIN *d*

Appends the domain name *d* to the local host name for authentication purposes. For example, if `gethostname()` returns the name IRIS, but the fully qualified domain name is IRIS.NENE.COM, you would use the DOMAIN option to set the domain to NENE.COM.

ESCAPE *xx,yy,...*

Escapes the specified characters on transmission (regardless of whether the peer requests them to be escaped with its async control character map). Specify the characters to be escaped as a list of hex numbers separated by commas. Note that you can specify almost any character for the ESCAPE option, unlike the ASYNCMAP option that only lets you specify control characters. The characters that cannot be escaped are those with hex values `0x20` through `0x3f`, and `0x5e`.

FILE *file*

Reads options from a file.

-IP

Disables IP address negotiation. If used, you must specify the remote IP address with an option on the command line, or in the TCPWARE:PPPOPTIONS.DAT file.

IPCP-ACCEPT-LOCAL

Accepts the peer's interpretation of the local IP address, even if the local IP address was specified in an option.

IPCP-ACCEPT-REMOTE

Accepts the peer's interpretation of its (remote) IP address, even if the remote IP address was specified in an option.

IPCP-MAX-CONFIGURE *n*

Sets the maximum number of IPCP configure-request transmissions to *n* (default 10).

IPCP-MAX-FAILURE *n*

Sets the maximum number of IPCP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

IPCP-MAX-TERMINATE *n*

Sets the maximum number of IPCP terminate-request transmissions to *n* (default 3).

IPCP-RESTART *n*

Sets the IPCP restart interval (retransmission timeout) to *n* seconds (default 3).

IPPARAM *string*

Provides an extra parameter to the IPUP and IPDOWN command file scripts. If used, the string supplied becomes the fifth parameter to those scripts.

IPUP *command-file*

Executes the specified command file when IP over PPP starts up, asynchronously, so that PPPD does not wait for the file's completion. Provided with the following command parameters:

Command Parameter	Description
P1	Interface name (such as PPP-0)
P2	Terminal device (such as _TTA3:)
P3	Local IP address
P4	Remote IP address
P5	(Optional) String specified with the IPPARAM option

If used with -D, debug output is logged in the *command-file*.LOG file.

IPDOWN *command-file*

Executes the specified file when IP over PPP shuts down, asynchronously, so that PPPD does not wait for the file's completion. Provided with the following command parameters:

Command Parameter	**Description**
P1	Interface name (such as PPP-0)
P2	Terminal device (such as _TTA3:)
P3	Local IP address
P4	Remote IP address
P5	(Optional) String specified with the IPPARAM option

If used with -D, debug output is logged in the *command-file*.LOG file.

KDEBUG *n*

Enables debugging of the low level interface with the IP and terminal driver. The argument *n* is a number that is the sum of the following values:

Value	**Description**
1	Enable general debug messages
2	Request that the contents of received PPP packets be printed
4	Request that the contents of transmitted PPP packets be printed
8	Request that raw data be received from the serial device
16	Request that raw data be transmitted to the serial device

LCP-ECHO-FAILURE *n*

Presumes the peer is dead if *n* LCP echo-requests are sent without receiving a valid LCP echo-reply. If this happens, PPPD terminates the connection. Requires a nonzero value for the LCP-ECHO-INTERVAL parameter. Use this option to enable PPPD to terminate after the physical connection breaks (such as when the modem hangs up) in situations where no hardware modem control lines are available.

LCP-ECHO-INTERVAL *n*

Sends an LCP echo-request frame to the peer every *n* seconds. Normally the peer should respond to the echo-request by sending an echo-reply. You can use this option with the LCP-ECHO-FAILURE option to detect when the peer is no longer connected.

LCP-MAX-CONFIGURE *n*

Sets the maximum number of LCP configure-request transmissions to *n* (default 10).

LCP-MAX-FAILURE *n*

Sets the maximum number of LCP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

LCP-MAX-TERMINATE *n*

Sets the maximum number of LCP terminate-request transmissions to *n* (default 3).

LCP-RESTART *n*

Sets the LCP restart interval (retransmission timeout) to *n* seconds (default 3).

LOGIN

Uses the system password database for authenticating the peer using PAP.

-MN

Disables magic number negotiation. With this option, PPPD cannot detect a looped-back line.

MRU *n*

Sets the MRU (Maximum Receive Unit) value to *n* for negotiation. The peer is requested to send packets of no more than *n* bytes. The minimum value is 128 and the default is 1500; 296 is recommended for slow links (40 bytes for the TCP/IP header plus 256 bytes of data).

-MRU

Disables MRU negotiation. PPPD uses the default MRU value of 1500 bytes.

MTU *n*

Sets the MTU (Maximum Transmit Unit) value to *n*. Unless the peer requests a smaller value through MRU negotiation, PPPD requests that the IP layer send data packets of no more than *n* bytes through the PPP network interface.

NAME *n*

Sets the name of the local system for authentication purposes to *n*.

NBDNS *address*

Identifies the primary NetBIOS name server.

NETMASK *n*

Sets the interface netmask to *n*, a 32 bit netmask in "decimal dot" notation (such as 255.255.252.0).

NOIPDEFAULT

Disables the default action when no local IP address is specified, which is to determine (if possible) the local IP address from the hostname. With this option, the peer must supply the local IP address during IPCP negotiation (unless you explicitly specify it on the command line, or in the TCPWARE:PPPDOPTIONS.DAT file).

+PAP

Requires the peer to authenticate itself using PAP.

-PAP

Disables authentication using PAP.

PAP-MAX-AUTHREQ *n*

Sets the maximum number of PAP authenticate-request transmissions to *n* (default 10).

PAP-RESTART *n*

Sets the PAP restart interval (retransmission timeout) to *n* seconds (default 3).

PAP-TIMEOUT *n*

Sets the maximum time that PPPD waits for the peer to authenticate itself with PAP to *n* seconds (0 means no limit).

PASSIVE
-P

Enables the "passive" option in the LCP. With this option, PPPD attempts to initiate a connection; if it does not receive a reply from the peer, it waits passively for a valid LCP packet from the peer (instead of exiting, as it does without this option).

-PC

Disables protocol field compression negotiation (the default).

PERSIST

Disables exiting after a connection is terminated. Instead, tries to reopen the connection.

PROXYARP

Adds an entry to this system's ARP (Address Resolution Protocol) table with the IP address of the peer and the Ethernet address of this system. This also starts the PPP interface as an unnumbered interface.

-PROXYARP

Disables the PROXYARP option. If you want to prevent users from creating proxy ARP entries with PPPD, place this option in the TCPWARE:PPPDOPTIONS.DAT file.

REMOTENAME *n*

Sets the assumed name of the remote system for authentication purposes to *n*.

SDNS *address*

Identifies the secondary DNS name server. If omitted, PPPD uses the first name server specified by the TCPWARE_NAMESERVERS logical, set up through Domain Name Services configuration.

SILENT

Disables transmitting LCP packets to initiate a connection until a valid LCP packet is received from the peer.

SNBDNS *address*

Identifies the secondary NetBIOS name server.

USEHOSTNAME

Enforces the use of the hostname as the name of the local system for authentication purposes (overrides the NAME option).

USER *u*

Sets the username to use to authenticate this machine with the peer using PAP to *u*.

-VJ

Disables negotiation of Van Jacobson style TCP/IP header compression (the default is to negotiate TCP/IP header compression).

-VJCCOMP

Disables the connection-ID compression option in Van Jacobson style TCP/IP header compression. With this option, PPPD does not omit the connection-ID byte from Van Jacobson compressed TCP/IP headers, nor request the peer to do so.

VJ-MAX-SLOTS *n*

Sets the number of connection slots to be used by the Van Jacobson TCP/IP header compression and decompression code to *n*, which must be between 2 and 16 (inclusive).

Troubleshooting PPPD

PPPD provides two types of debugging information:

- Trace output to SYS$OUTPUT
- OPCOM messages

By default, PPPD generates OPCOM messages for fatal errors, such as a failure to open the IP interface or insufficient privileges. In normal running operation, you should not see any OPCOM message. If you have a problem getting PPPD to work, first search for the OPCOM messages for PPPD.

You can also specify the DEBUG option. This enables the PPPD process to print out informational messages to SYS$OUTPUT. Define SYS$OUTPUT to the appropriate log file before invoking the PPPD server (or you can invoke PPPD interactively and output to the terminal). You must specify -DETACH to use this option.

When you specify the DEBUG (or -D) option, it debugs at level 5, which is to display up to warning and significant events. For more informational and debugging information, you can raise the debug level up to 7 by defining the logical TCPWARE_PPPD_DEBUG_LEVEL.

For a detached process, or if you prefer, you can also raise the message level for OPCOM messages. By default, it is set to 4 to report fatal and error messages. You may want to raise it to 5 to monitor the significant events in PPPD, or even higher for more detail by defining the logical TCPWARE_PPPD_OPCOM_LEVEL.

If you get the OPCOM messages:

```
%TCPware_PPPD-E-setting terminal device failed with error 0x2C4
%TCPware_PPPD-E-PPP device initialization failed with error 0x2C4
%SYSTEM-F-DEVACTIVE, device is active,
```

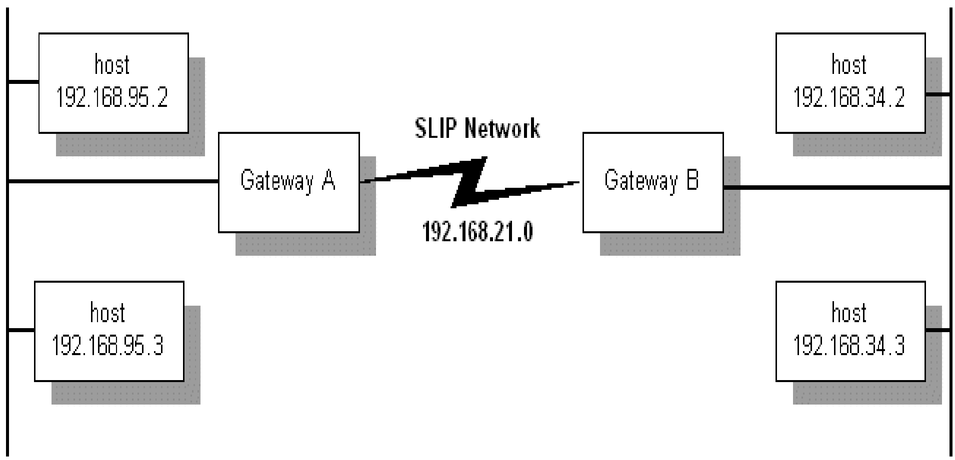
Make sure that the *device-name* indicated on the PPPD command line is allocated to the current process before starting PPPD.

Serial Line IP Interface

Use serial Line IP (SLIP) when you need to route TCP/IP traffic over a serial line instead of an Ethernet cable. You most commonly use SLIP to connect systems on two Ethernet networks some distance apart.

You need TCP/IP and application software at each end of the SLIP link. One end can be TCPware, while the other end can be a SLIP implementation such as ULTRIX or SunOS. Figure 4-5 shows a typical SLIP network.

Figure 4-5 Sample Internet with SLIP Link



Because SLIP lines are point-to-point connections, with TCPware you have the option to configure SLIP lines as network lines or as unnumbered interface lines.

TCPware supports both dedicated and dialup SLIP lines. Configure dedicated (hard-wired) SLIP lines during network configuration. Configure dialup SLIP lines as you need them, as described in this chapter. SLIP is common with line speeds from 1200 bits per second (bps) to 19.2 Kbps.

NETCP (not IPDRIVER) does the I/O to the terminal device to send and receive datagrams. NETCP uses the IPDRIVER External Interface to do this.

SLIP Line Identification

You can use any standard OpenVMS terminal device as a SLIP line. Unlike other line ID controller numbers, the SLIP line ID is not related to the actual device name. CNFNET prompts you for the actual device name during TCPware configuration.

The START/IP command *line-specific-information* parameter provides the OpenVMS device name for the SLIP line. If you omit this parameter, TCPware assumes that the TCPWARE_ SLIP_*n* system logical (where *n* is the controller number) defines the device.

The maximum number of SLIP lines you can configure for one TCPware host is 256. You can define lines SLIP-0 through SLIP-255. If you try to define a SLIP line with a larger number in CNFNET, the message `%TCPWARE_CNFNET-E-INVLINE, invalid line` appears.

Before Configuring SLIP Lines

Before you begin configuring SLIP lines:

- You must set the TTY_ALTYPAHD system parameter larger than its default value. This avoids losing characters. The higher the line speed, the higher you should set this parameter. For most applications, 1024 is appropriate.

  TTY_ALTYPAHD is not a dynamic parameter. If you use SYSGEN to change it, you must reboot the system for the change to take effect.
- The MAXBUF parameter must be at least twice the maximum transmission unit (MTU) of the SLIP line, plus 144. The default MTU for SLIP lines is 1006 bytes; therefore, MAXBUF must be at least 2156. Increase MAXBUF if necessary. MAXBUF is dynamic. If you use SYSGEN to change it, you do not need to reboot the system for the change in value to take effect.

Configuring SLIP Lines

To configure TCPware for SLIP:

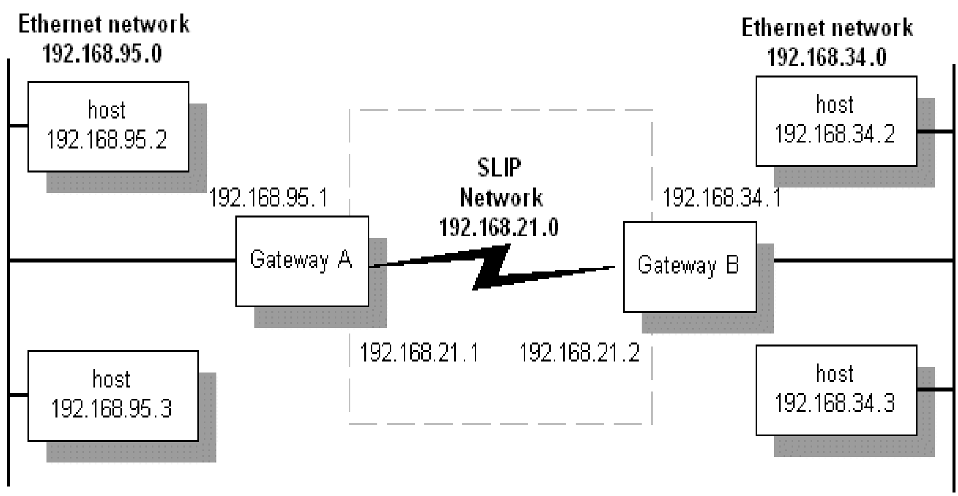
1	If you plan to use a dedicated SLIP line, enter its line ID, host name, internet address, and terminal device name in response to the applicable prompts in CNFNET. You can use any valid OpenVMS terminal device as a SLIP line. CNFNET prompts you for the actual device name. Make sure that the network number portion of the SLIP line's internet address is unique if you use the /UNNUMBERED interface flag.
2	For dedicated SLIP lines, you may want to create the TCPWARE:SLIP_SETUP.COM file. The network startup command procedure (TCPWARE:STARTNET) executes this command procedure, if it exists, before using the SLIP lines. SLIP_SETUP.COM should contain the commands necessary to configure the terminal devices for proper operation. Typically, it would include SET TERMINAL commands to set the baud rate and other terminal characteristics.

3	For SLIP line speeds higher than 1200 bps, enable the alternate type-ahead buffer (ALTYPEAHD) characteristic for the terminal. Enter the following command for each SLIP terminal at the DCL prompt or in the SLIP_SETUP.COM file: **`SET TERMINAL /ALTYPEAHD /PERMANENT`** ***`device`***
4	For both dedicated and dialup SLIP lines, set up routing information so that TCP/IP traffic routes properly over the SLIP link. The SLIP link should either have a unique network number or be unnumbered. You can give TCPware routing information either in the Network Control Utility (NETCU), by editing the TCPWARE:ROUTING.COM file to include appropriate NETCU commands, or through GATED. For example, enter the following commands on each SLIP terminal at the DCL prompt or in the SLIP_SETUP.COM file (for dedicated lines): **`ADD ROUTE /NETWORK`** ***`network-address`*** **`/GATEWAY`** ***`gateway-address`*** **`ENABLE FORWARDING`** NETCU entries remain active until TCPware shuts down. Updating the ROUTING.COM file with these commands makes them permanent. Do not use this method if using GateD to configure routes. To use GateD to configure routes, include a `static` statement for each of the routes in the TCPWARE:GATED.CONF file. (See the next section.)

Sample SLIP Link

Figure 4-6 shows a sample internet consisting of three networks: Ethernet network 192.168.95.0, SLIP network 192.168.21.0, and Ethernet network 192.168.34.0.

Figure 4-6 Sample Internet with SLIP Link



Each gateway has an internet address for each network to which it connects. In this example, you can do the following to set up SLIP network 192.168.21.0 so that networks 192.168.95.0 and 192.168.34.0 can communicate:

- At each TCPware host in network 192.168.95.0 (on the Gateway A side), set the local gateway host address:
  `SET GATEWAY 192.168.95.1`
- Do the same for each host in network 192.168.34.0 (on the Gateway B side):
  `SET GATEWAY 192.168.34.1`
- At Gateway A, add the route through Gateway B's SLIP network address:
  ```
  ADD ROUTE /NETWORK 192.168.34.0 /GATEWAY 192.168.21.2
  ENABLE FORWARDING
  ```
- At Gateway B, add the route through Gateway A's SLIP address:
  ```
  ADD ROUTE /NETWORK 192.168.95.0 /GATEWAY 192.168.21.1
  ENABLE FORWARDING
  ```

Note! You can also define the default gateway by responding to prompts during the network configuration procedure. See Chapter 3, *Configuring the TCP/IP Core Environment*, in the *Installation & Configuration Guide*.

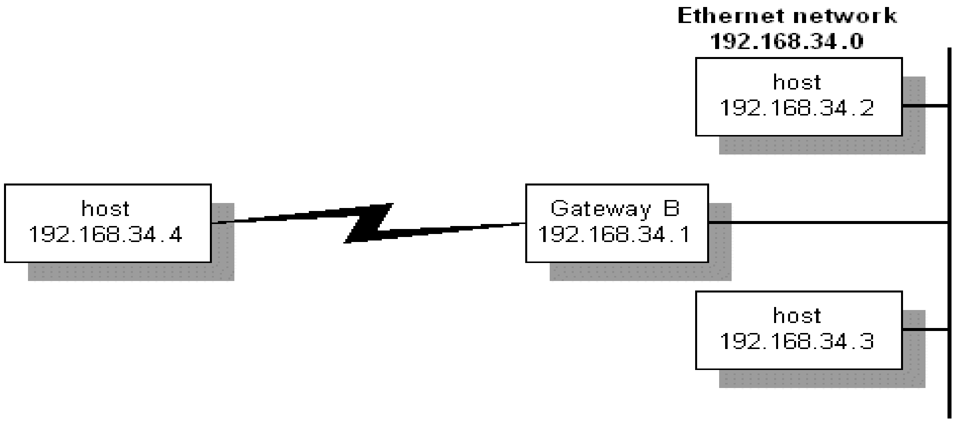
You can also configure the SLIP route using GateD. Include the following statements in the GATED.CONF files instead of the ADD ROUTE commands in ROUTING.COM:

```
static
{ 192.168.34.0 gateway 192.168.21.2 ;} ;
static
{ 192.168.95.0 gateway 192.168.21.1 ;} ;
```

Sample Unnumbered SLIP Link

In the case of a single host connection over a SLIP line to a network, you may not be able to dedicate a separate SLIP address to the connection on the network end. This requires you to use an unnumbered interface. Figure 4-7 shows such a scenario.

Figure 4-7 Sample SLIP Connection with an Unnumbered Interface



Here is how you would set up the connection on host 192.168.34.4:

```
SET GATEWAY 192.168.34.1
```

Here is how you would set up the connection at Gateway B for host 192.168.34.2:

```
START/IP /UNNUMBERED SLIP-0 192.168.34.1
ADD ROUTE 192.168.34.4 SLIP-0
ENABLE FORWARDING /ARP
```

You need the /UNNUMBERED qualifier with the START/IP command. The ENABLE FORWARDING command with the /ARP qualifier lets host 192.168.34.1 respond to Address Resolution Protocol (ARP) requests for the remote host's address. In this way, other hosts on the 192.168.34.0 network can send any packets addressed to *new remote ip addr* to Gateway B. This is known as "proxy ARP" and keeps you from having to add the host route on all other hosts in network 192.168.34.0.

You can also set up unnumbered interfaces on both ends of the SLIP connection. You must use ENABLE FORWARDING, but you cannot use proxy ARP with unnumbered interfaces, and you have to add routes on all other nodes in the network.

Incoming Dialup SLIP Lines

Perform these steps to set up an incoming dialup SLIP line:

1	Create a login account for the SLIP site using the OpenVMS AUTHORIZE utility. This should be a captive account and must have either OPER privilege or have been granted the TCPWARE_CONTROL rights identifier. TCPWARE_CONTROL requires the software password file, PSW_*.DAT, to have at least read access for the TCPWARE_CONTROL rights identifier.

2	Create a LOGIN.COM file for this account. A template file is available in TCPWARE:SLIPLOGIN.COM. Example 6-1 shows an unnumbered interface example.

In the example, a user is dialing in from a PC configured to use an address of 192.168.95.124. The Ethernet interface on the OpenVMS system is configured with an address of 192.168.95.12.

Example 4-1 Sample LOGIN.COM File

```
$ ON WARNING THEN LOGOUT
$ IF (F$TRNLNM("TCPWARE_NETCP_MBX") .EQS. "") THEN GOTO EXIT
$ NETCU := $TCPWARE:NETCU
$ TT = F$TRNLNM("TT")
$ DEFINE/USER SYS$ERROR NLA0:
$ DEFINE/USER SYS$OUTPUT NLA0:
$ NETCU START/IP/UNNUMBERED SLIP-1 192.168.95.12 'TT'
$ IF ($STATUS .NE. 1) THEN GOTO EXIT
$ NETCU ADD ROUTE 192.168.95.124 SLIP-1
$ NETCU ENABLE GATEWAY/ARP
$ EXIT:
$ LOGOUT
```

See *Sample Unnumbered SLIP Link* for details on setting unnumbered interfaces and using proxy ARP.

Once you set up the account and login file, the remote site simply dials up the OpenVMS system and logs in as the SLIP user to establish the connection. The login command file automatically configures the SLIP line.

Make sure to create a separate account and LOGIN.COM file for each remote SLIP site.

Outgoing Dialup SLIP Lines

To set up an outgoing dialup SLIP line:

1	Allocate the terminal device you wish to use. Enter: **`ALLOCATE`** ***`terminal`***
2	Set the terminal characteristics (such as the baud rate) using SET TERMINAL commands.
3	Use SET HOST/DTE, KERMIT, or some other utility to dial the remote system and log in as the SLIP user.
4	If the remote end successfully starts the SLIP line, exit SET HOST/DTE or KERMIT.

5	Start the outgoing SLIP line. Enter: **`NETCU START/IP`** ***`SLIP-unit internet-address terminal`*** • SLIP-*unit*--is the controller number of the SLIP line you want to assign (this number is for identification only and must be unique). You can use an asterisk (*) as a wildcard value, which assigns the lowest unused line id to the SLIP interface (starting with SLIP-0), and also defines the TCPWARE_LINE (global) symbol to be that interface. (See the START/IP command in the *NETCU Command Reference*, Chapter 2, *NETCU Commands*.) • *internet-address*--is the internet address of the local host for the SLIP network. • *terminal*--is the terminal device name. You can also add any of the NETCU START/IP qualifiers supported for SLIP lines on the NETCU START/IP line. The following is a sample outgoing SLIP line startup command: **`NETCU START/IP SLIP-0 192.168.95.6 TXA7`**
6	Deallocate the terminal device. Enter: **`DEALLOCATE`** ***`terminal`*** The SLIP line is now ready to use.

Disconnecting SLIP Lines

To disconnect a SLIP line, enter:

`NETCU STOP/IP` ***`SLIP-unit`***

Note! TCPWare automatically removes the SLIP line from the network configuration if you configure the terminal device as a modem line with hangup enabled, and you lose the phone line for any reason.

Full XON/XOFF Flowcontrol

The /FLAGS=FLOWCONTROL qualifier with the START/IP command configures the OpenVMS terminal device for full XON/XOFF control (READSYNC, HOSTSYNC, and TTSYNC). This means that you can use high speed modems that support compression and reliable data transfer modes. In addition, when OpenVMS terminal devices use full flow control, they do not need to use the alternate type-ahead buffer.

When you use /FLAGS=FLOWCONTROL qualifier with the START/IP command, TCPware configures the SLIP line to run a modified SLIP protocol. The modified SLIP protocol maps the characters shown in Table 4-10. Note that all numeric values are in octal.

Table 4-10 SLIP Protocol Character Mapping

Character name...	Has ASCII value...	With mapped character sequence...	
SLIP End of packet	300	333	334

Table 4-10 SLIP Protocol Character Mapping

SLIP Escape	333	333	335
XON	021	333	336
XON + 200	221	333	337
XOFF	023	333	340
XOFF + 200	223	333	341

RFC 1055, *A Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP*, defines the SLIP End-of-Packet and Escape characters, but does not define the XON/XOFF character mapping. TCPware uses this character mapping only when you specify /FLAGS=FLOWCONTROL.

Note! Only use /FLAGS=FLOWCONTROL when the other end of the SLIP line connects to a system running TCPware, and if you configure it to use this option.

Qualifiers with SLIP Lines

The START/IP command supports a number of other qualifiers that you can use with SLIP lines. For details, see Table 2-11 and Table 2-14 in the *NETCU Command Reference*.

Compressed SLIP

Use compressed SLIP (CSLIP) to compress the TCP/IP headers only (and not the data) over the SLIP line.

You can set CSLIP options in NETCU for the serial line to either compress all TCP/IP headers or to compress them if it receives a compressed header from the peer. Use the /FLAGS qualifier for the START/IP command, as follows:

This Command...	Specifies that the serial line should compress...
/FLAGS=COMPRESSED	all TCP/IP headers.
/FLAGS=AUTOENABLE	TCP/IP headers only if the peer sends compressed TCP/IP headers.

See Table 2-14 of the START/IP command reference in the *NETCU Command Reference*.

Troubleshooting SLIP

Access error messages help by entering **`HELP TCPWARE MESSAGES`** ***`[identifier]`***, or connect to web site **`http://www.process.com`** (select **`Customer Support`** followed by the **`Error Messages`** button).

Also keep the following in mind:

- If you are not running TCPware on both ends of the SLIP connection, avoid using XON/XOFF flow control with SLIP. If you have a modem that uses XON/XOFF, disable that mechanism.
- If SLIP performance is poor, check that you configured the terminal to use the alternate type-ahead buffer (using SET TERMINAL /ALTYPEAHD /PERMANENT), and that you adjusted the SYSGEN TTY_ALTYPAHD parameter.

Chapter 5

Cluster Alias Failover

Introduction

This chapter describes how to implement cluster alias failover for your VMScluster.

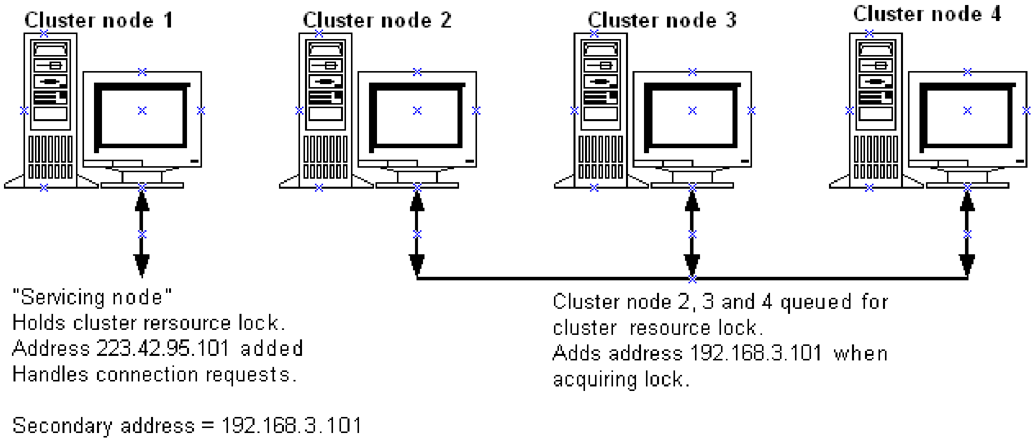
Cluster alias failover allows you to set up an alias node to provide continued connection in case a system fails. The alias accepts incoming connection requests in a VMScluster for a server if the servicing node goes down. Use cluster alias failover primarily for NFS over UDP, but you can also use it for other TCP/IP protocols such as FTP or TELNET.

For cluster alias failover to work properly, make sure that you have the same set of directories exported from each VMScluster node. If it works effectively, cluster alias failover allows users to continue working productively if cluster nodes go down.

How It Works

Each VMScluster node has a unique internet address. With cluster alias failover, you can assign a common secondary internet address, or alias. This alias is always an Internet address and never a name. This same secondary address is given to each VMScluster node that handles connection requests, and is recognized as a local address (see Figure 5-1).

Figure 5-1   Cluster Alias Failover Operation



One of the nodes accepts the OpenVMS cluster-wide resource lock, adds the alias, and handles the incoming connection requests. Other VMScluster nodes are also assigned the alias queue for the resource lock. If the servicing node goes down (or you shut down TCPware on it), the system releases the resource lock. One of the queued VMScluster nodes acquires the resource lock and adds the secondary address.

CAUTION! Do not use cluster alias addresses as Domain Name Services addresses.

Setting It Up

Use the ADD SECONDARY command in the *NETCU Command Reference* to set up cluster failover. For example:

```
NETCU> ADD SECONDARY 192.168.3.101
```

The address 192.168.3.101 becomes the local alias address for the interface.

You can include the /CLUSTER_LOCK qualifier with the ADD SECONDARY command. This qualifier instructs the VMScluster node to accept the OpenVMS cluster-wide resource lock before adding the secondary address. If another node in the VMScluster holds the lock, the node queues for the lock and adds the address when it acquires it.

Use the REMOVE SECONDARY command in the *NETCU Command Reference* to remove an alias added through ADD SECONDARY. If the system holds a cluster lock, use the /ABORT qualifier to force removal of the secondary address.

Be sure to add the ADD SECONDARY command to the TCPware ROUTING.COM file so that it can take effect each time you start TCPware for OpenVMS.

Limitations

There is no concept of a primary node with cluster alias failover. The alias address only moves to another address when the active servicing node goes down. The alias does not go back to the original servicing node when it comes back up.

You can move the alias address to a particular node by issuing the REMOVE SECONDARY /ABORT command in NETCU, as follows:

1	You can also add any of the NETCU START/IP qualifiers supported for SLIP lines on the NETCU START/IP line. The following is a sample outgoing SLIP line startup command: `NETCU START/IP SLIP-0 192.168.95.6 TXA7`
2	This time do a REMOVE SECONDARY /ABORT on the node currently holding the lock. The desired node now holds the lock.
3	Do an ADD SECONDARY on all other nodes. Using cluster alias failover has several other limitations: • Cluster alias failover does not interact with dynamic TCP/IP load balancing (Chapter 1). • TCP connections, such as TELNET and FTP, go down if the servicing node crashes.

Chapter 6

Managing SNMP Services

Introduction

This chapter explains the following Simple Network Management Protocol (SNMP) information:

- Links and traps
- Management Information Base (MIB)
- Configuring the SNMP Services
- Maintaining the SNMP configuration file
- Extendible MIB support
- SNMP Multiplexing (SMUX) peers support
- Log file

SNMP Services allows network management stations to obtain timely information about the network activities of OpenVMS server hosts. The information describes such things as routing, line status, the volume of network traffic, and error conditions.

Links

In SNMP, network communication lines are called *links*. When counting the number of IP datagrams sent and received over most links, the SNMP agent returns the same numbers that are available through the SHOW NETWORKS command in TCPware's Network Control Utility (NETCU). These numbers indicate how many datagrams TCPware delivers.

Traps

A trap is an unsolicited message the SNMP agent sends to a management station to inform it that a change in the network occurred. The management station is responsible for diagnosing and monitoring any reported problems. For example, the SNMP agent sends traps to tell the management stations which communication lines are running and which are down.

The SNMP agent sends traps only to clients configured to receive traps, as defined in the SNMP agent configuration file (SNMPD.CONF, described in the *SNMP Configuration File* section). The SNMP agent supports all traps defined in the SNMP protocol, except EGP-Neighbor-Loss, Warm-Start, and Enterprise-Specific.

TCPware initially enables all supported traps. If for any reason you may want to disable them, you can do so by editing the SNMP agent configuration file (see *Traps*). The changes take effect the next time you start the agent.

SNMP clients can enable or disable Authentication Failure Traps while the SNMP agent is running. These clients must have READ-WRITE community access, as described in the *MIB Access Rules* section.

Management Information Base

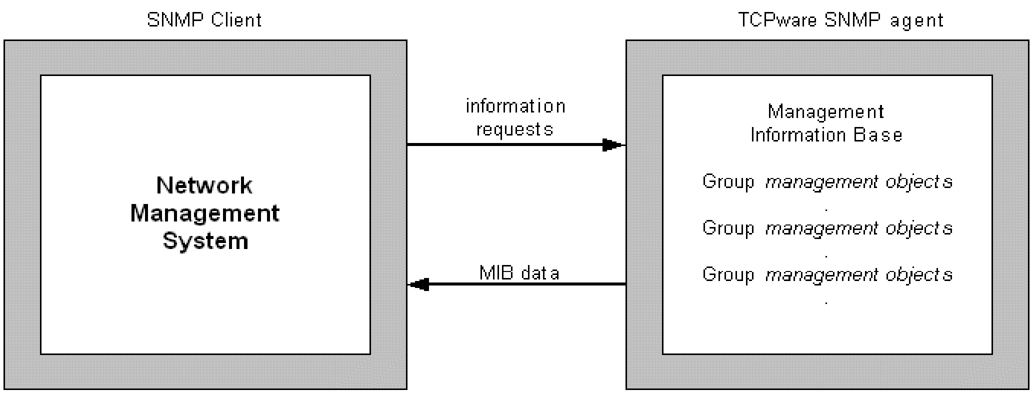
A Management Information Base (MIB) is a collection of network management data residing on the SNMP agent host. The network management station reads and writes MIB data to the agent. Related types of data in the MIB are in groups. Each piece of data within a group is a *management object*.

All management objects in a MIB are coded in ASN.1. Any authorized clients (or those using the community name "public") can access data in the MIB by using the SNMP Get and GetNext requests.

MIB-II is the MIB version for TCP/IP implementations. The SNMP agent supports all management objects defined in MIB-II, except those in the External Gateway Protocol (EGP) Group.

Figure 6-1 shows an SNMP client and agent exchanging MIB data.

Figure 6-1 Exchanging MIB Data



MIB Access Rules

Two kinds of rules restrict access to the MIB:

- Community access profiles
- The access mode assigned to each management object – NONE, READ-ONLY, READ-WRITE, and WRITE-ONLY. The SNMP protocol standard determines the access mode.

The network administrator assigns each SNMP agent and client to at least one community. A community consists of SNMP agents and clients that have the same *access profile*, or collection of rules that determine whether community members can:

- Read or write MIB data
- Receive traps

You define access profiles in the SNMP agent configuration file.

Clients with READ-WRITE community access can alter the values of certain management objects in the MIB.

MIB Groups

Table 6-1 summarizes the information in each MIB group.

See also RFC 1213, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*, for complete information on each MIB group.

Table 6-1 Summary of MIB Data Group

Group	Contains objects...	Which...
System	sysDescr sysUpTime sysContact sysName sysLocation sysServices	Provides information about the agent host, such as the domain name, geographic location, and the name of a contact person.

Table 6-1 Summary of MIB Data Group (Continued)

Group	Contains objects...	Which...
Interfaces	ifNumber ifTable ifEntry ifIndex ifDescr ifType ifMtu ifSpeed ifPhysAddress ifAdminStatus ifOperStatus ifLastChange ifInOctets ifInUcastPkts ifInNUcastPkts ifInDiscards ifInErrors ifInUnknownProtos ifOutOctets ifOutUcastPkts ifOutNUcastPkts ifOutDiscards ifOutErrors ifOutQLen ifSpecific	Provides generic information about each network interface, such as the speed, administrative status, and the maximum size of transmission units. Counts the number of data errors, and the number of packets sent and received. Contains the Interfaces Table.

Table 6-1 Summary of MIB Data Group (Continued)

Group	**Contains objects...**	**Which...**
Address Translation (AT)	atTable atEntry atIfIndex atPhysAddress atNetAddress	Maps the network (IP) address to the physical address.

Table 6-1 Summary of MIB Data Group (Continued)

Group	Contains objects...	Which...
IP	ipForwarding ipDefaultTTL ipInReceives ipInHdrErrors ipInAddrErrors ipForwDatagrams ipInUnknownProtos ipInDiscards ipInDelivers ipOutRequests ipOutDiscards ipOutNoRoutes ipReasmTimeout ipReasmReqds ipReasmOKs ipReasmFails ipFragOKs ipFragFails ipFragCreates ipAddrTable ipAddrEntry ipAdEntAddr ipAdEntIfIndex ipAdEntNetMask ipAdEntBcastAddr ipAdEntReasmMaxSize ipRouteTable ipRoute Entry ipRouteInfo	Counts the number of datagrams sent, received, in error, discarded, fragmented, and reassembled. Contains the IP Address Table, IP Routing Table, and IP Address Translation Table.

Table 6-1 Summary of MIB Data Group (Continued)

Group	Contains objects...	Which...
	ipRouteIfIndex ipRouteMetric1 ipRouteMetric2 ipRouteMetric3 ipRouteMetric4 ipRouteNextHop ipRouteType ipRouteProto ipRouteAge ipRouteMask ipRouteMetric5 ipRouteInfo ipNetToMediaTable ipNetToMediaEntry ipNetToMediaIFIndex ipNetToMediaPhyAddress ipNetToMediaNetAddress ipNetToMediaType ipRoutingDiscards	

Table 6-1 Summary of MIB Data Group (Continued)

Group	**Contains objects...**	**Which...**
ICMP	icmpInMsgs icmpInErrors icmpInDestUnreachs icmpInTimeExcds icmpInProbs icmpInSrchQuenchs icmpInRedirects icmpInEchos icmpInEchoReps icmpInTimestamps icmpInTimestampReps icmpInAddrMasks icmpInAddrMaskReps icmpOutMsgs icmpOutErrors icmpOutDestUnreachs icmpOutTimeExcds icmpOutParmProbs icmpOutSrcQuenchs icmpOutRedirects icmpOutEchos icmpOutEchoReps icmpOutTimestamps icmpOutTimestampReps icmpOutAddrMasks icmpOutAddrMaskReps	Counts the number of ICMP messages sent, received, and in error. Also counts source quenches, redirects, and timestamps.

Table 6-1 Summary of MIB Data Group (Continued)

Group	Contains objects...	Which...
TCP	tcpRtoAlgorithm tcpRtoMin tcpRtoMax tcpMaxConn tcpActiveOpens tcpPassiveOpens tcpAttemptFails tcpEstabResets tcpCurrEstab tcpInSegs tcpOutSegs tcpRetransSegs tcpConnTable tcpConnEntry tcpConnState tcpConnLocalAddress tcpConnLocalPort tcpConnRemAddress tcpConnRemPort tcpInErrs tcpOutRsts	Counts the number of active opens, passive opens, and failed attempts. Also contains the TCP Connection Table.

Table 6-1 Summary of MIB Data Group (Continued)

Group	Contains objects...	Which...
UDP	udpInDatagrams udpNoPorts udpInErrors udpOutDatagrams udpTable udpEntry udpLocalAddress udpLocalPort	Counts the number of datagrams sent and received. Also contains the UDP Listener Table.

Table 6-1 Summary of MIB Data Group (Continued)

Group	Contains objects...	Which...
SNMP	snmpInPkts snmpOutPkts snmpInBadVersions snmpInBadCommunityNames snmpInBadCommunityUses snmpInASNParseErrs snmpInTooBigs snmpInNoSuchNames snmpInBadValues snmpInReadOnlys snmpInGenErrs snmpInTotalReqVars snmpInTotalSetVars snmpInGetRequests snmpInGetNexts snmpInSetRequets snmpInGetResponses snmpInTraps snmpOutTooBigs snmpOutNoSuchNames snmpOutBadValues snmpOutGenErrs snmpOutGetRequests snmpOutGetNexts snmpOutSetRequests snmpOutGetResponses snmpOutTraps snmpEnableAuthenTraps	Counts the number of packets sent and received, invalid community names, and invalid version numbers, and SNMP errors. Also counts the number of requests, responses, and traps sent and received.

Configuring SNMP Services

To configure SNMP Services, follow these steps:

1	Invoke the CNFNET procedure by entering the following command at the DCL prompt: `$ TCPWARE:CNFNET SNMP`
2	Edit the SNMP configuration file, as described in the next section.
3	Restart TCPware.

Configuration File

The SNMP configuration file is SNMPD.CONF. The TCPWARE_ROOT directory includes this file.

The SNMP configuration file defines:

- Values for a subset of MIB management objects.
- Clients and communities who can access the SNMP agent.
- MIB access privileges for each client and community.
- Authentication Failure, Link Up, and Link Down traps' status.
- SMUX peer details

Note! After editing the configuration, stop and restart the SNMP agent so that the changes can take effect.

If you do not edit the configuration file, the SNMP agent uses default values.

File Format

Follow these guidelines when entering data in the SNMP configuration file:

- Allow one line for each item.
- Enter information in any order; in upper- or lowercase.
- Enter variable string information (*id-string* and *contact-name*) in upper- or lowercase, depending on the operating system. Some SNMP clients in your network (such as those running UNIX) may require information in a specific case.
- Place quotation marks (" ") around strings that contain spaces or that occupy more than one line in the file.
- Use a pound sign (#) or an exclamation point (!) to denote comments. SNMP ignores all information following these characters. It treats the pound sign and exclamation point like regular characters if they appear within quotation marks ("").

Values for MIB Objects

To define the values of several MIB objects in the SNMP configuration file, use the corresponding keywords listed in Table 6-2.

Table 6-2 Management Objects

MIB object name...	Has keyword...
system.sysDescr	SYSDESCR
system.sysContact	SYSCONTACT
system.sysLocation	SYSLOCATION
if.ifTable.ifEntry.ifDescr and if.ifTable.ifEntry.ifSpeed	INTERFACE
system.sysServices	SYSSERVICES

The following paragraphs explain how you define each item.

SYSDESCR *[id-string]*

The *id-string* should include the full name of the hardware, operating system, and networking software. For example:

```
SYSDESCR "AlphaServer 8400, VMS V7.1, Process Software Corporation TCPware
for OpenVMS"
```

If you omit the *id-string*, TCPware tries to obtain this information from your current system. If the attempt fails, the default is `System description is unknown`.

SYSCONTACT *[contact-name]*

The *contact-name* specifies the person to contact for the host, and how you can contact this person (such as by mailbox address). For example:

```
SYSCONTACT "John Smith, X 1234, smith@process.com"
```

The default is `System contact is unknown at this time`.

SYSLOCATION *[system-location]*

The *system-location* specifies the geographical location of the host. For example:

```
SYSLOCATION "959 Concord Street, Framingham, MA"
```

The default is: `System location is unknown at this time`.

INTERFACE *[line-id line-speed description]*

The *line-id* specifies the line identification for the IP layer network device. The *line-speed* specifies the line speed in bits per second. The *description* specifies the manufacturer's name, product name, and hardware version for the interface. For example:

```
INTERFACE qna-1 10000000 "DELQA Ethernet Controller Version 1.0"
```

If you do not enter a description, TCPware tries to obtain one from your current system. If the attempt fails, the default is `System description is unknown`.

SYSSERVICES *services-set-number*

The SNMP agent uses a default value of 72 for this MIB object. You can override this value in the configuration file.

RFC 1213, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*, explains how to calculate the value of *services-set-number.*

Community Parameters

The SNMP configuration file must contain the following information for each client permitted access to the SNMP agent:

COMMUNITY *community-name internet-address type*

community-name	Specifies the name of the community to which the client belongs. This parameter is case-sensitive.
internet-address	Specifies the client's internet address. If you enter 0.0.0.0, any address can use the community.
type	defines the access profile as one of the following: • **READ-ONLY**--The client can retrieve data from the MIB on this host • **READ-WRITE**--The client can retrieve data from and write data to the MIB on this host • **TRAPS**--The client will receive all enabled traps

COMMUNITY public 0.0.0.0 is always READ-ONLY. Do not enter any other definition for it. The SNMP agent ignores it.

Example 6-1 shows some community parameters defined in the configuration file.

Example 6-1 Community Parameters

```
community northeast  192.168.4.56  READ-ONLY
community northeast  192.168.220.1 READ-WRITE
community southwest  192.168.23.1  READ-WRITE
community southwest  192.168.23.1  TRAPS
```

- Client 192.168.4.56 in the NORTHEAST community has READ-ONLY access to the MIB, while client 192.168.220.1 in the same community has READ-WRITE access.
- Client 192.168.23.1 belongs to the SOUTHWEST community. This community has READ-WRITE access to the MIB and can receive all traps.

Disabling Traps

All traps that the SNMP agent supports are initially enabled. You can disable traps by editing the configuration file. These changes take effect the next time you start the agent. Table 6-3 shows how to disable traps.

Table 6-3 Disabling Traps

Disable this trap...	By entering...
Authentication Failure	no-auth-traps
Link Up	no-link-traps
Link Down	no-link-traps

Note! SNMP clients can enable or disable the Authentication Failure Trap while the SNMP agent is running. These clients must have READ-WRITE community access.

SNMP Multiplexing Peers

The SNMP Multiplexing (SMUX) protocol is an SNMP subagent extension protocol. Each subagent or peer registers a MIB subtree with the SNMP Agent. Requests for objects residing in a registered MIB subtree are passed from the SNMP Agent using the SMUX protocol to the subagent. The subagent passes the results of an SNMP query back to the SNMP Agent. The practical limit to the number of peers is 30.

The SNMP server only accepts SMUX connections from peers listed by IP address in the SNMPD.CONF file. Use the following syntax in the file:

SMUX_PEER *ip-address*

The SNMP agent listens on TCP port 199 for peer connections, while the connection to the SNMP client is over UDP port 161, with traps sent over UDP port 162. Multiple peers registering the same subtree are each assigned a priority, and the agent can send multiple variables in a single request.

The SMUX protocol is described in RFC 1227.

Template Configuration File

SNMP Services provides a TEMPLATE_SNMPD.CONF file in TCPWARE_COMMON:[TCPWARE] that you can use as a basis (see Example 6-2).

Example 6-2   Sample SNMP Configuration File

```
! SNMP Agent (SNMPD) Configuration File (template)
!
! System description: sysdescr <id string>
! Typically the id string would include:
! VAX cpu model (such as MicroVAX II, VAX 8650, etc)
! VMS and version number
! "Process Software Corporation, TCPware for OpenVMS Version 5.4"
!
sysdescr "place system description string here"
!

! System Contact: syscontact <contact name>
!
syscontact "place name, phone number, and mail address of administrator
here"
!
! System Location: syslocation <location>
!
syslocation "place system location information here"

! Line Interfaces Information: interface <line-id> <line speed>
!<description>
! Note: You usually need not define these. SNMPD provides good defaults.
!
!interface una-0 10000000 "COMPAQ DELUA Ethernet controller"
!
! Communities:
! community <community name> <internet address>
! <READ-ONLY | READ-WRITE | TRAPS>
!
community readers   192.168.1.2   READ-ONLY
community netman    192.168.2.3   READ-WRITE
community nettraps  192.168.3.4   TRAPS
!
! To disable authentication traps, remove the "!" from the following
! line.
!no-auth-traps
!
! To disable link status traps, remove the "!" from the following
```

```
1 line.
!no-link-traps
!
! SMUX Peers:
! SMUX_PEER <ip-address>
!
SMUX_PEER 192.168.4.5
SMUX_PEER 192.168.5.6
```

Private MIB Application Program Interface

In addition to SMUX, TCPware's SNMP agent supports subagents serving private MIBs through an application programming interface (API). Under this scheme, anyone willing to have their private MIBs served by TCPware's SNMP agent should develop a shareable image that exports the APIs in them in addition to the routines they may need for accessing the MIB variables.

The SNMP API routines are described in Chapter 18 of the *Programmer's Guide, SNMP Extendible Agent API Routines*.

SNMP Log File

When the SNMP agent starts up, it creates a log file called TCPWARE:SNMPSERVER.LOG. This file contains information about the activities of the SNMP agent, such as:

- The time the agent starts up and shuts down.
- When SMUX peers open or close a connection, and register or de-register a MIB tree.
- Any errors found in the SNMP configuration file.
- Any errors that occur when the agent is running.

Chapter 7

X.25 Interface

Introduction

This chapter describes the rules for adding information to the X25.CONF file so that you can build the X.25 mapping database. Review this entire chapter before attempting to do so. Examples of X25.CONF files and X25 module characteristics are included.

The chapter is divided into the following sections:

- X.25 support
- System parameters
- Addresses
- Mapping database
- Sample X25.CONF files
- Sample X.25 module characteristics
- Troubleshooting

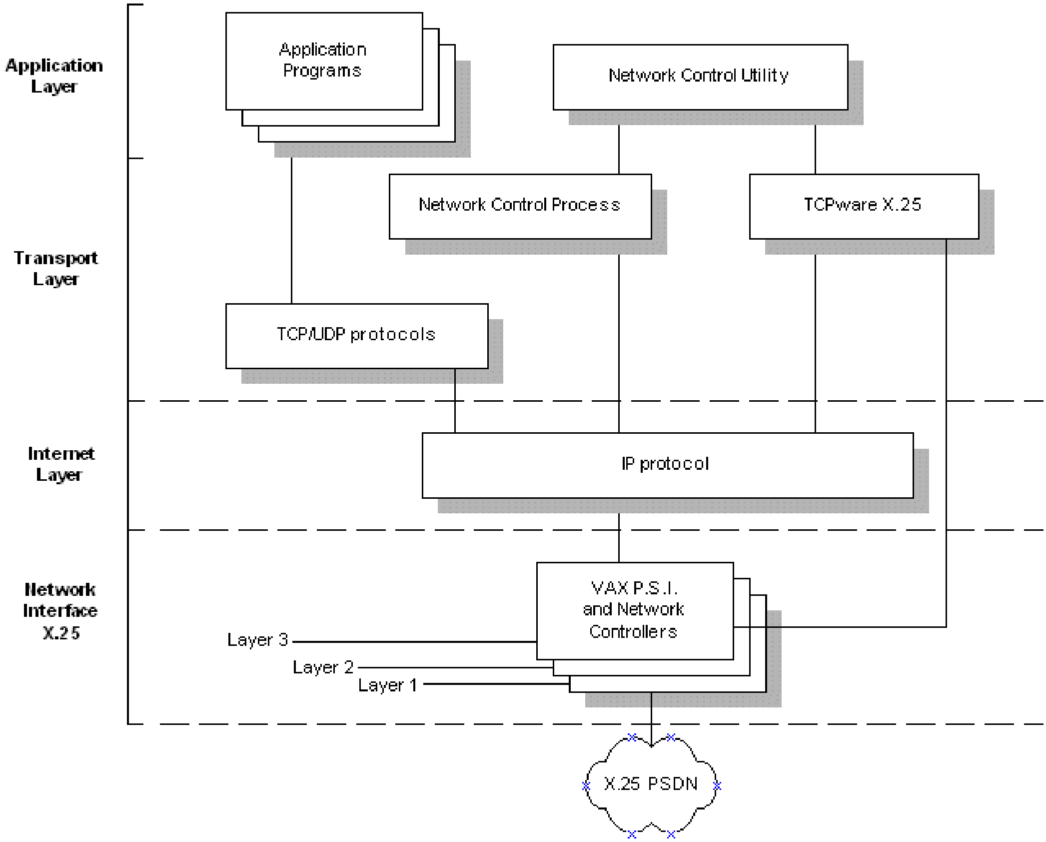
Support

TCPware X.25 interface support conforms to RFC 1356. You can send IP datagrams as data packets over an X.25 network. In this way you can send and receive error free and reliable communications worldwide. Figure 7-1 shows the interaction between TCPware and the X.25 protocols.

System Parameters

You must install VAX P.S.I. Version 4.3 or later on the system to provide IP-over-X.25 support. Compaq Computer now distributes another VAX P.S.I. (V5.0) with the DECnet-VAX Extensions. The DECnet-VAX Extensions are for VMS Version 5.4 and later.

Figure 7-1 Interaction Between TCPware and X.25 Protocols



You may need to change parameters for X.25 support using the following Network Control Utility (NETCU) commands:

	Command	Description
1	START/IP /MTU	For X.25 devices, set the maximum transmission unit (MTU) to 1500 bytes. However, if the system communicates with older versions of IP-over-X.25 (RFC 877), set the MTU to 576 bytes. Set the MTU to more than 1500 only by prearrangement with the other sites.

2	START/TCP / NOKEEPALIVE	Process Software recommends you set the /NOKEEPALIVE qualifier with the START/TCP command if anyone in your organization is on a tariff network. To disable KEEPALIVE, edit the TCPWARE:ROUTING.COM file and add: **`START/TCP /MSS=16384 /MWS=24576 /NOKEEPALIVE`**
3	SET TCPRTOMIN	Set the minimum TCP transmission time (TCPRTOMIN) to 2000 milliseconds for X.25 devices.
4	SET TCPRTOMAX	Set the maximum TCP transmission time (TCPRTOMAX) to 15000 milliseconds for X.25 devices.

Changes to network parameters made by these NETCU commands are active until TCPware shuts down. Update the ROUTING.COM file with these commands to make them permanent.

See the *NETCU Command Reference*, Chapter 2, *NETCU Commands*.

DTE or X.25 Addresses

Each local and remote host in an X.25 network has a unique DTE address that the network uses to route calls. This address is similar to a telephone number; for example, 31446172353970. The address is usually 14 digits long and referred to here as an X.25 address.

The first four digits, the Data Network Identification Code (DNIC), identify the country and the network within the country. The following example shows the DNIC in **bold**:

31446172353970

Vendors that supply the X.25 network service within the country, assign the remaining digits (frequently referred to as the National Address). The National Address usually consists of 10 digits, but can be more or less. The maximum number of digits allowed in an X.25 address is 15. Occasionally you can add an optional subaddress (usually two digits) to the end of the national number.

Table 7-1 shows how one vendor assigns the national address based on geographic area codes. Other vendors use other schemes to assign the national address for a DTE.

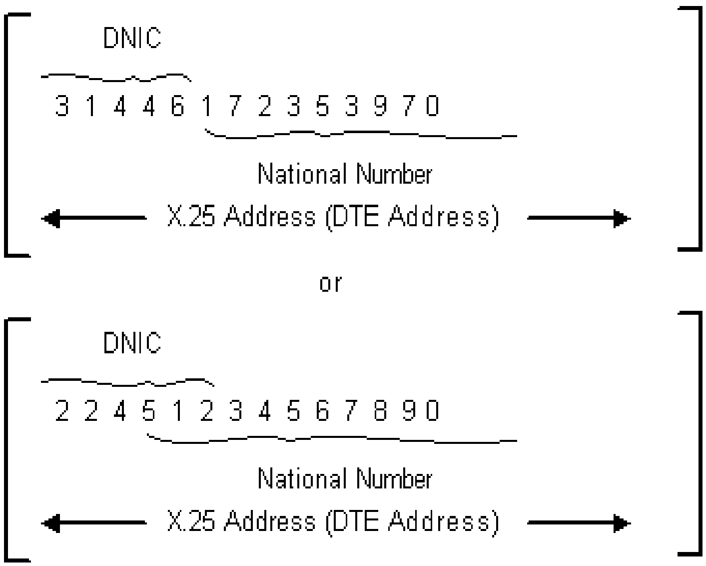
Table 7-1 Structure of an X.25 Address

	National Address		
DNIC	**Country geographic code**	**Local geographic code**	**Specific DTE address**
3144	617	235	3970
3144	603	952	3698

Also, each host in IP-over-X.25 networking has an IP address; for example, 192.168.95.68.

Figure 7-2 shows two sample X.25 addresses. The vendor in the second example uses the last digit of the DNIC as part of the national number.

Figure 7-2   Two X.25 Addresses



You must use both the X.25 address and IP addresses of the remote host when you create entries in the X25.CONF file. The X25.CONF file always resides on the local host.

Mapping Database

Subscribers using X.25 network service must maintain a database table that maps the Internet and X.25 addresses of remote hosts. The system manager builds or changes this table by entering commands in the X25.CONF file.

The contents of the X25.CONF file provide the necessary information for a local host (Data Terminal Equipment [DTE]) to communicate with a remote DTE over an X.25 network.

Table 7-2 lists the commands you use to build the mapping database in the X25.CONF file. Figure 7-3 shows the relationship of the command entries to the remote DTE. Table 7-2 lists the commands you use to build the mapping database in the X25.CONF file. Figure 7-3 shows the relationship of the command entries to the remote DTE.

Table 7-2 Commands to Build the X.25 Mapping Database

Command	Does the following...
map	Maps the IP address to a network name. The map command line defines the IP address and the network name of the remote host and, if the remote host is on a network, the IP mask. Its syntax is described in the *Adding Map Entries* section.
route	Maps the network name to a host name. The route command line defines the network name and host name of the remote host. Its syntax is described in the *Adding Route Entries* section.
address	Maps the host name to an actual call number. The address command line defines the host name, if communication is through X.25 or ISDN, and the X.25 or ISDN address (full DTE call number) of the remote host. Its syntax is described in the *Adding Address Entries* section.
translate	Maps the X.25 address (full DTE call number) to the digits used in the local carrier network. The translate command line identifies the local carrier, and defines how to translate the digits of the X.25 address to the digits that actually go over the local carrier network. Its syntax is described in the *Adding Translate Entries* section. Translate entries contain information about the local carrier network; all other entries contain information about the remote host.

Figure 7-3 Mapping to a Remote DTE (Host)

```
Host -------------->   IP --->   Hostname   --->  Route  --->  X.25 address  -->        Actual call
Information             |            |                |         ISDN address      |
                        v            v                v                           v
X25.CONF-------> map        route        address                   translate
database
commands
```

Table 7-3 Commands to Build the X.25 Mapping Database

Command	Does the following...
map	Maps the IP address to a network name. The map command line defines the IP address and the network name of the remote host and, if the remote host is on a network, the IP mask. Its syntax is described in the *Adding Map Entries* section.
route	Maps the network name to a host name. The route command line defines the network name and host name of the remote host. Its syntax is described in the *Adding Route Entries* section.

Table 7-3 Commands to Build the X.25 Mapping Database (Continued)

Command	Does the following...
address	Maps the host name to an actual call number. The address command line defines the host name, if communication is through X.25 or ISDN, and the X.25 or ISDN address (full DTE call number) of the remote host. Its syntax is described in the *Adding Address Entries* section.
translate	Maps the X.25 address (full DTE call number) to the digits used in the local carrier network. The translate command line identifies the local carrier, and defines how to translate the digits of the X.25 address to the digits that actually go over the local carrier network. Its syntax is described in the *Adding Translate Entries* section. Translate entries contain information about the local carrier network; all other entries contain information about the remote host.

Before You Begin

Before you add mapping information to the X25.CONF file:

1	Make sure you are connected to X.25 through the Direct Access Facility (DAF).
2	Obtain from the remote network managers: • Name and IP address of the remote hosts • Network name of the remote hosts • X.25 address for the remote hosts (a call number and Data Network Identification Code [DNIC])e host(s)
3	Obtain from your local carrier: • Name of the local carrier entered during configuration of Compaq's VAX P.S.I. software; for example, PSC or TRANSPAC • Number sequence (escape codes) the local carriers uses to translate the X.25 address to the digits that go over the local carrier network

See also Appendix B, *Data Network Identification Codes.*

Database

Table 7-2 on the previous page describes the function of all the X.25 mapping database commands. To build the X.25 mapping database in the X25.CONF file, do the following:

1	Create the SYS$SPECIFIC:[TCPWARE]X25.CONF file: **a** Add all map entries associated with the first remote host. Use the syntax: **`map`** ***`IP address network-name [mask IP-mask]`*** **b** Add all route entries associated with the first remote host. Use the syntax: **`route`** ***`network-name host-name`*** **c** Add all address entries associated with the first remote host. Use the syntax: **`address`** ***`host-name`*** **`{`*****`X.25 | ISDN`*****`}`** ***`address`*** **d** Enter the translate entry associated with the first carrier. Use the syntax: **translate {*X.25 | ISDN*} *pattern PSI-network-name pattern***
2	Repeat step 1 for each additional carrier connected to the local host.
3	Save the X25.CONF file and exit the editor.

Tips

The following subsections step you through building a sample mapping database in the X25.CONF file. Example 7-1 shows the full X25.CONF file.

Tips for building your mapping database in the X25.CONF file include:

- Enter a full set of mapping commands (as listed in Table 7-2 for each remote host you want to communicate with over the X.25 network; for example, map, route, address, and translate.
- Each mapping command you enter in the X25.CONF file has the explicit syntax described in the following subsections.
- The X25.CONF file resides on the local host.
- Translate entries contain information about the local carrier network; all other entries contain information about the remote host.
- You can have multiple map, route, address, and translate entries. The order that you enter these commands in the file does not matter.

Example 7-1 shows the X25.CONF file built using the examples shown in the next subsections. PSC is the local carrier. In this example, the local host can communicate with the remote hosts `eta.process.com`, `delta.process.com`, and `henri.napole.com`.

Example 7-1 Sample X25.CONF File

```
map 192.168.95.1 eta.nene.com
map 192.168.95.68 nene-net.nene.com 255.255.255.0
map 192.168.10.0 big.cahuna.com 255.255.0.0
```

```
route nene-net.nene.com delta.nene.com
route big.chauna.com henri.cahuna.com

address eta.nene.com X.25 31446172353980
address delta.nene.com X.25 31446172353970
address henri.cahuha.com X.25 20801234567890

translate X.25 3144* PSC *
translate X.25 * PSC 1*
```

Map Entries

The map entry maps the IP address of the remote host to the network name on which the remote host resides.

Create map entries for the remote hosts in the X25.CONF file as shown in Example 7-2. See also the map and related entries highlighted in Figure 7-4.

If you map to the whole network, include the IP mask.

Example 7-2 Syntax of Map

`map` ***`IP-address network-name [mask IP-mask]`***

For example:

```
map 192.168.95.68 nene-net.nene.com mask 255.255.255.0
```

or

```
map 192.168.10.0 big.cahuna.com mask 255.255.0.0
```

As shown in Example 7-3, you do not need to enter a network mask if you add a map entry for a single system.

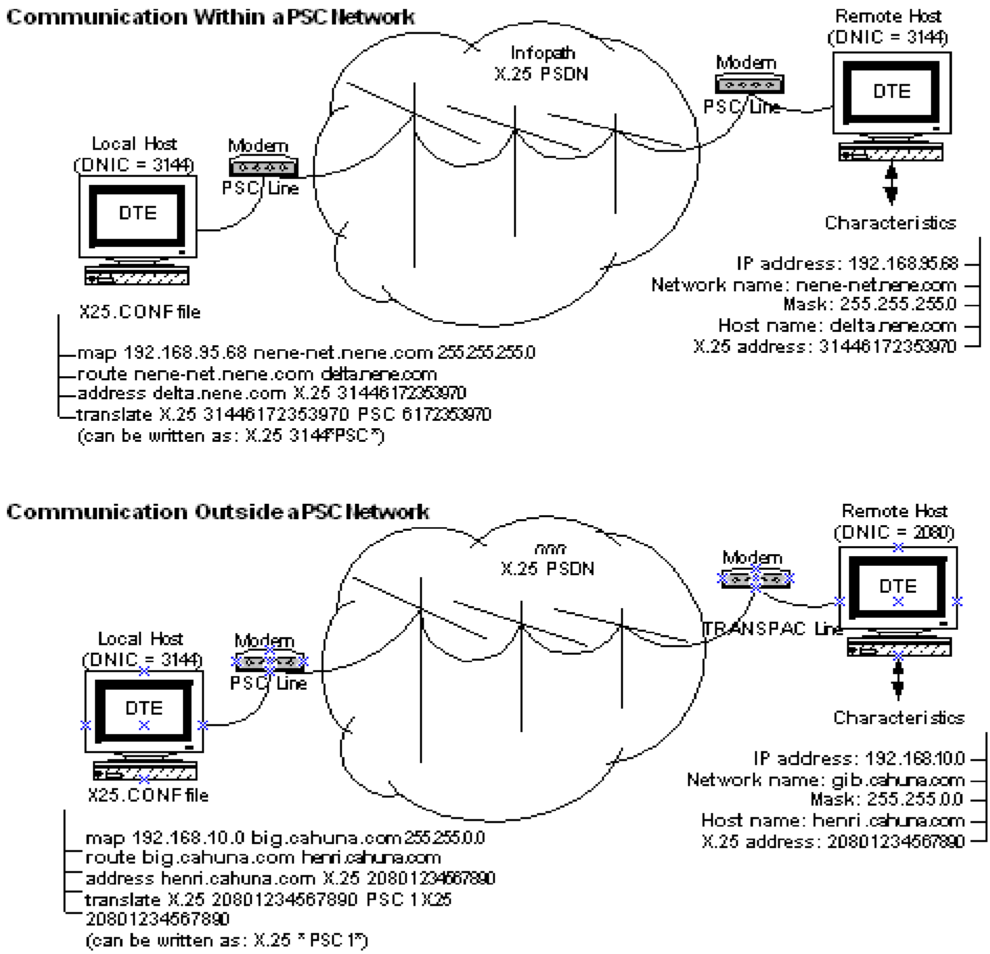
Example 7-3 Mapping to a Single System

`map` ***`IP-address host-name`***

For example:

```
map 192.168.95.1 eta.nene.com
```

Figure 7-4 Building an X25.CONF File: Map



Route Entries

The route entry maps the network name of the remote network to the name of the remote host.

Create route entries for the remote hosts in the X25.CONF file as shown in Example 7-4. See also the route and related entries highlighted in Figure 7-5.

Example 7-4 Syntax of Route

`route` ***`network-name host-name`***

For example:

```
route nene-net.nene.com delta.nene.com
```

or

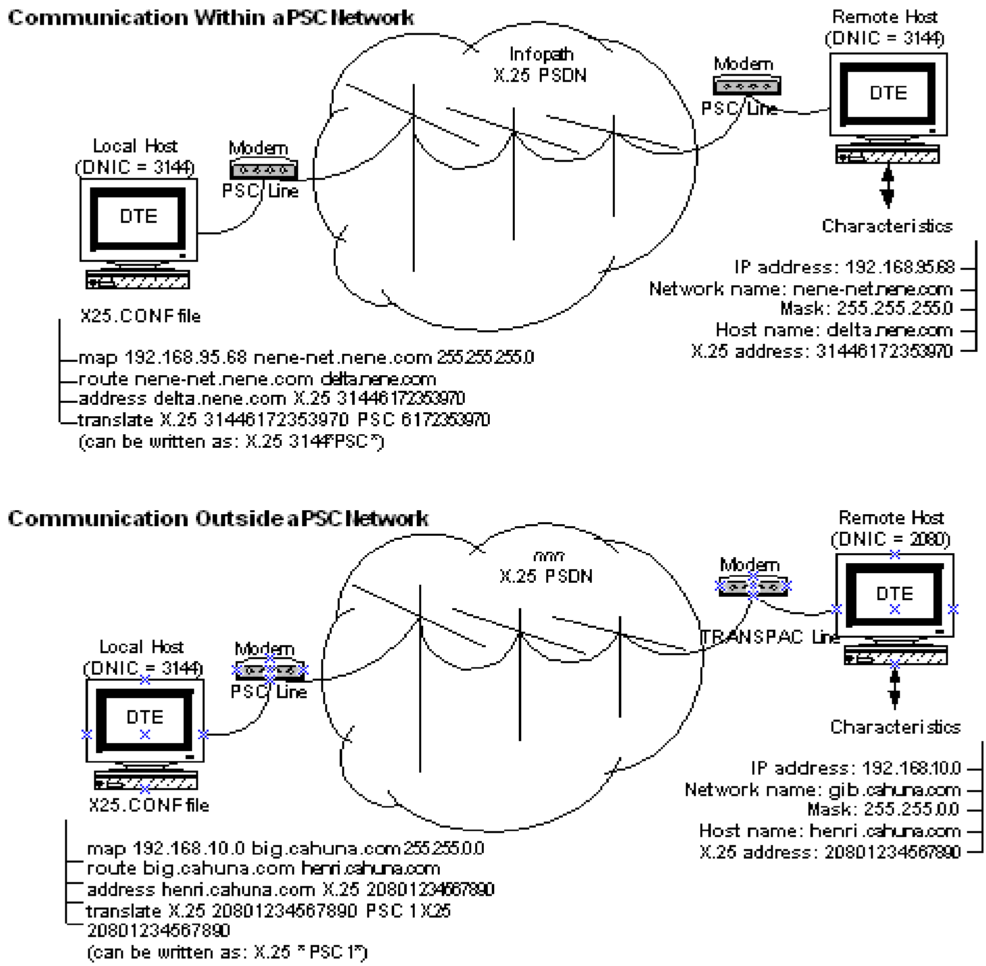
```
route big.cahuna.com henri.cahuna.com
```

You do not need to include route information if the host routes to itself. For example, no route information need accompany the entry

```
map 192.168.95.1 eta.nene.com
```

shown in Example 7-3.

Figure 7-5 Building an X25.CONF File: Route



Address Entries

The address entry maps the name of the remote host to the X25 address for the remote host.

Create address entries for the remote hosts in the X25.CONF file as show in Example 7-5. See also address and related entries shown in Figure 7-6.

Note that the *address* is the X.25 address. It includes the DNIC and the national address. You can also specify optional X.25 standard facilities as shown in Example 7-5.

Example 7-5 Syntax of Address

```
address host-name {ISDN | X.25} address FACILITY <hex-string>]

                    [WINDOW number]
                    [PACKET number]
                    [REVERSE]
                    [INCOMING]
```

For example:

`address delta.nene.com X.25 31446172353970`

or

`address eta.nene.com X.25 31446172353980` (not shown in Figure 7-6)

or

`address henri.cahuna.com X.25 20801234567890`

In this example, the hex string is local facilities added to the call, including the PSDN-specific number if needed. Use the options shown in the example for facilities such as packet size, window, or reverse charging.

Note! The `INCOMING` option means that you should only use the address to recognize incoming calls and never to make outgoing calls.

See Appendix B, *Data Network Identification Codes*, for a listing of DNICs and related information. Contact your local carrier for information if your long distance carrier is not listed, or the necessary information is not included in the table.

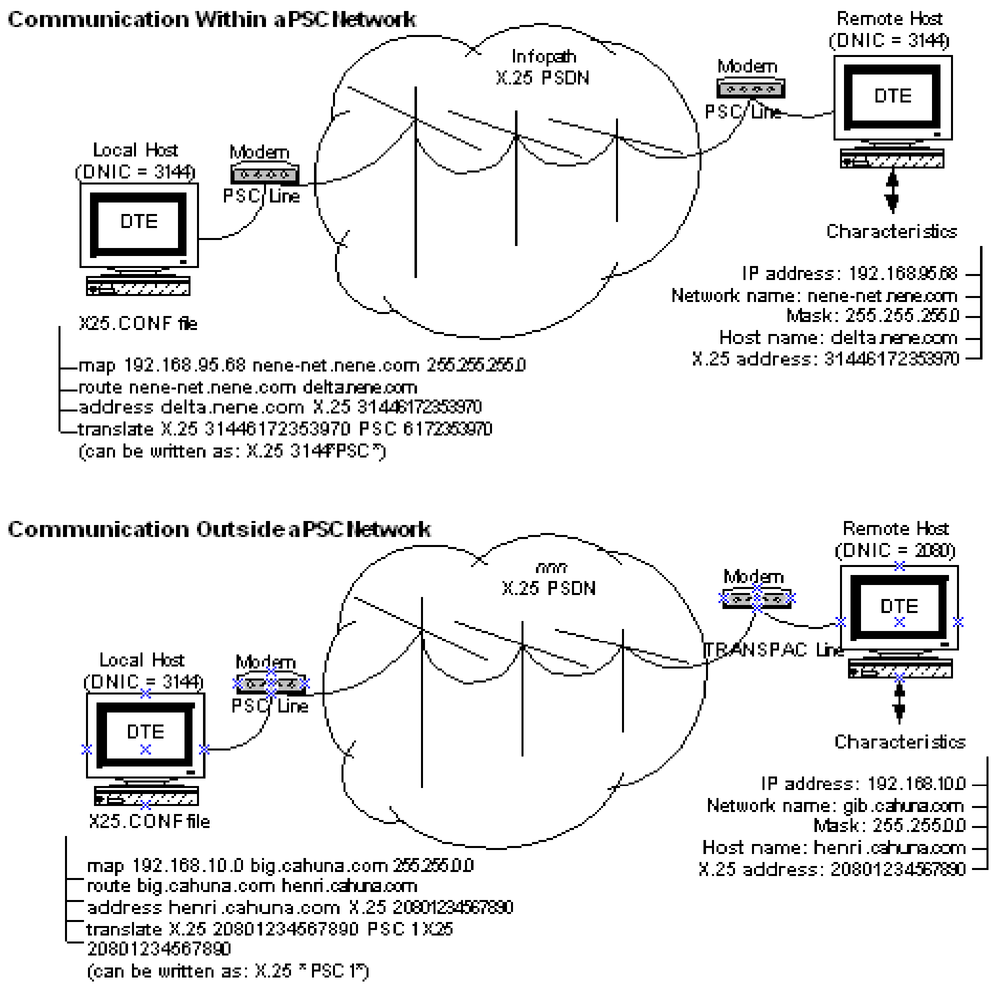
Translate Entries

The translate entry maps the X25 address of the remote host to the digits used in the local carrier network. There are two types of translate entries that you may need to include in the X25.CONF file:

- Entries for communication within the local carrier network
- Entries for communication going outside the local carrier network

Each carrier has its own set of rules governing the translation of X.25 and ISDN addresses to the digits that actually go over the carrier network. See Appendix B, *Data Network Identification Codes*, or contact your local carrier for this information.

Figure 7-6 Building an X25.CONF File: Address



Within a Carrier Network

Create the translate entries for communication within the carrier network as shown in Example 7-6. See also the translate and related entries shown in Figure 7-7.

Example 7-6 Syntax of Translate Within a Carrier Network

translate {X.25 | ISDN} pattern PSI-network-name pattern

where the first *pattern* is the X.25 address (DTE address) and second *pattern* is the number string required by the local carrier network.

For example:

`translate X.25 3144* PSC *` (see **Note 1**)

or

`translate X.25 2080* TRANSPAC *` (see **Note 1**)

or

`translate X.25 2145* IBERPAC 5*` (see **Note 2**; not shown in Figure 13-7)

Note 1: For any X.25 communication going from the local host: If the DNIC portion of the destination X.25 address matches the DNIC of the local carrier (3114 or 2080), then remove all the DNIC digits and keep all the digits represented by the asterisk (*). Send this string over the local carrier network.

Note 2: For any X.25 communication going from the local host: If the DNIC portion of the destination X.25 address matches the DNIC of the local carrier, then remove the first three DNIC digits (in this case 214) and keep the last DNIC digit (in this case 5) and all the digits represented by the asterisk (*). Send this string over the local carrier network.

See Appendix B, *Data Network Identification Codes*, for details. Contact your local carrier if the necessary *pattern* information is missing.

Outside a Carrier Network

Create the translate entries for communication outside the carrier network as shown in Example 7-7. See also the translate and related entries shown in Figure 7-8.

In this example, you must specify the full X.25 address prefixed by "1" to communicate with a remote host located in a different carrier network.

Example 7-7 Syntax of Translate Outside a Carrier Network

translate {*X.25 | ISDN*} *pattern PSI-network-name pattern*

where the first *pattern* is the X.25 address and second *pattern* is the number string required by the local carrier network.

For example:

`translate X.25 * PSC 1*`

For any X.25 communication going from the local host: If the DNIC portion of the destination X.25 address does not match any of the other translate entries, then add a 1 in front of all the digits represented by the *. Send this string over the local carrier network.

See also Appendix B, *Data Network Identification Codes*, for details. Contact your local carrier if the necessary *pattern* information is missing.

Figure 7-7 Building an X25.CONF File: Translate (Within a Network)

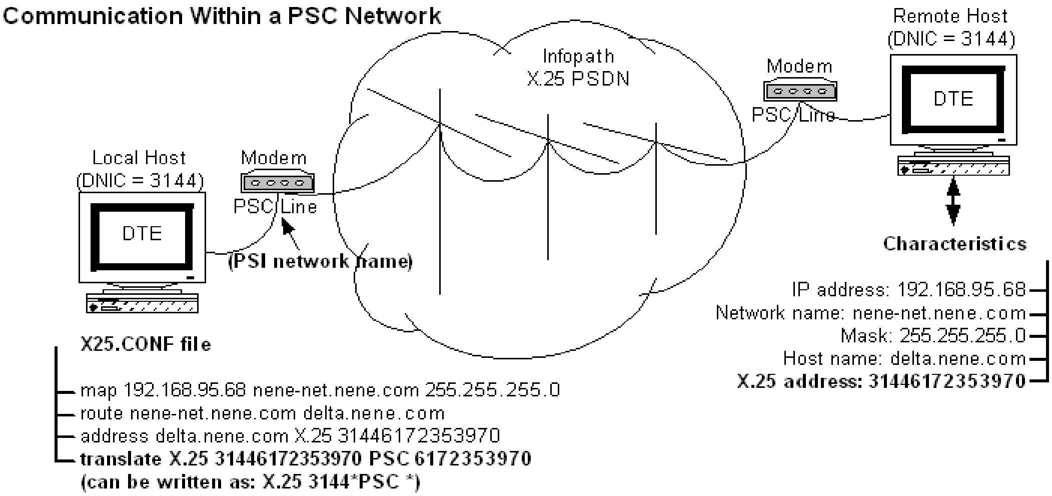
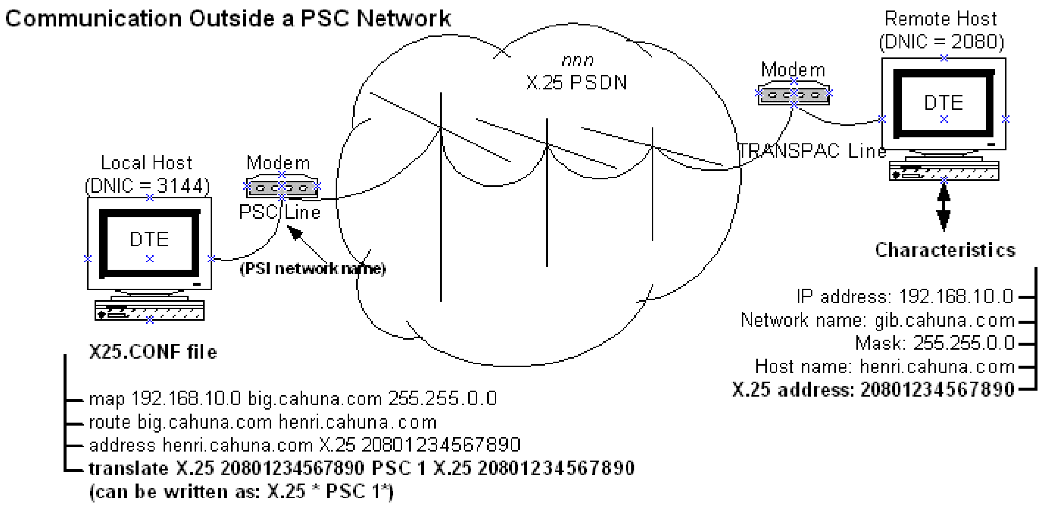


Figure 7-8 Building an X25.CONF File: Translate (Outside a Network)



Sample X25.CONF Files

The following examples show the X25.CONF file for three host systems, Condor, Eagle, and Hawk. Condor and Hawk subscribe to the carrier network DCS in Belgium. Eagle subscribes to carrier network DATANET in the Netherlands.

Table 7-3 shows the characteristics for each host.

Table 7-4 Characteristics of a Condor, Eagle, and Hawk

Characteristics...	For Condor...	For Eagle...	For Hawk...
IP address	192.168.1.9	192.168.1.7	192.168.3.1
Network name	calif.ert.com	aviary.environ.co m	sierra.green.com
Mask	255.255.255.0	255.255.0.0	none
Host name	condor.ert.com	eagle.environ.com	hawk.green.com
X25 address	2062555121	2041345678	2062765443
Carrier network	DCS	DATANET	DCS

The carrier network DCS has a DNIC of 2062. The translate entry for this DNIC is:

`translate X.25 2062* DCS 2*`
(remove 206 from the string but retain 2 for communication withing the DCS network)

`translate X.25 * DCS 0*` (add 0 to the string for communication outside the network)

The carrier network DATANET has a DNIC of 2041. The translate entry for this DNIC is:

`translate X.25 2041* DCS 1*`
(remove 204 from the string but retain 1 for communication within the DCS network)

`translate X.25 * DCS 0*` (add 0 to the string for communication outside the network)

On Condor

In Example 7-8, the local host Condor can communicate with the remote hosts `eagle.environ.com` and `hawk.green.com`.

Because `hawk.green.com` subscribes to the DCS carrier network, Condor uses the first translate entry to establish communication with Hawk.

Because `eagle.environ.com` does not subscribe to the DCS carrier network, Condor uses the second translate entry to establish communication with Eagle.

Example 7-8 X25.CONF File on Condor

```
map 192.168.1.7 aviary.environ.com 255.255.0.0
route aviary.environ.com eagle.environ.com
address eagle.environ.com X.25 2041345678

map 192.168.3.1 sierra.green.com
route sierra.green.com hawk.green.com
address hawk.green.com X.25 2062765443

translate X.25 2062* DCS 2*
translate X.25 * DCS 0*
```

On Hawk

In Example 7-9 the local host Hawk can communicate with the remote hosts `eagle.environ.com` and `condor.ert.com`.

Because `condor.ert.com` subscribes to the DCS carrier network, Hawk uses the first translate entry to establish communication with Condor.

Because `eagle.environ.com` does not subscribe to the DCS carrier network, Hawk uses the second translate entry to establish communication with Eagle.

Example 7-9 X25.CONF File on Hawk

```
map 192.168.1.7 aviary.environ.com 255.255.0.0
route aviary.environ.com eagle.environ.com
address eagle.environ.com X.25 2041345678

map 192.168.3.1 calif.ert.com
route calif.ert.com condor.ert.com
address condor.ert.com X.25 2062555121

translate X.25 2062* DCS 2*
translate X.25 * DCS 0*
```

On Eagle

In Example 7-10 the local host Eagle can communicate with the remote hosts `condor.ert.com` and `hawk.green.com`.

Because neither `condor.ert.com` nor `hawk.green.com` subscribe to the DATANET carrier network, Eagle uses the second translate entry to establish communication with Condor and Hawk.

Example 7-10 X25.CONF File on Eagle

```
map 192.168.3.1 calif.ert.com
route calif.ert.com condor.ert.com
address condor.ert.com X.25 2062555121

map 192.168.3.1 sierra.green.com
route sierra.green.com hawk.green.com
address hawk.green.com X.25 2062765443

translate X.25 2041* DATANET 1*
translate X.25 * DATANET 0*
```

Sample Module Characteristics

Example 7-11 shows the NCP command and output to demonstrate the X.25 module characteristics on the specified DTE. If you use X.25 access, use the NCP command that starts with **`SHOW MODULE X25 ACCESS ...`** to display the network characteristics.

Example 7-11 DTE 1: X25 Module Characteristics

```
NCP> show module x25-p known dtes

Module X25-Protocol Volatile Summary as of 1-Jun-1997 13:03:49
```

DTE	Network	State	Active Channels	Active Switched
990002	DIRECT	on - unsynchronized	0	0
6172353970	PSC	on - running	2	2

```
NCP> show module X25-p dte 990002 characteristics

Module X25-Protocol Volatile Characteristics as of 1-Jun-1999 13:-04:25
```

Example 7-12 shows the NCP command and output to demonstrate the X.25 module characteristics on the specified DTE.

Example 7-12 NCP Command

```
NCP> show module X25-p dte 6172353970 characteristics

Module X25-Protocol Volatile Characteristics as of 1-Jun-1999 13:-04:52
```

```
DTE = 6172353970
Network = PSC
Line = DSV-0-0
Channels = 4-1
Maximum channels = 4
Maximum circuits = 512
Default data = 128
Default window = 2
Maximum data = 128
Maximum window = 2
Maximum clears = 2
Maximum resets = 2
Call timer = 200
Clear timer = 180
Reset timer = 180
Restart timer = 180
Interrupt timer = 180
Interface mode = DTE
```

Troubleshooting

If you have problems running X.25, do the following:

- Enter the **TRACE *** command as the first line in the X25.CONF file on the OpenVMS system.

  This causes an analysis of what occurred on the network to be written to the X25.LOG file. You do not have to restart TCPware after you edit the X25.CONF file. Note that each new X.25 connection reads this file.

- If you have VAX P.S.I. with X.29, enter the SET HOST/X29 command to try to connect to your destination.

  This commands uses DECnet and VAX P.S.I. to connect rather than TCPware. If this fails, there is a problem with your P.S.I. configuration and TCPware will not work until the problem is corrected.

- Enter NCP, then enter the following command:

  ```
  NCP> SHOW MODULE X25-ACCESS KNOWN NETWORKS
  ```

  This NCP command shows the names of X.25 networks as they are configured in VAX P.S.I. This command is useful when there is a possibility that the X25.CONF file might be specifying a national X.25 network incorrectly.

  TCPware and VAX P.S.I. need to be configured to use the same network name. Sometimes VAX P.S.I. might be configured with nicknames for the national networks. Later you might get a different name for the network from your carrier and enter that in the X25.CONF file. In this case you need to change the X25.CONF file or reconfigure VAX P.S.I.

PART III Managing Routing

Chapter 8

Routing and GateD

Introduction

This chapter describes TCPware's multiple gateway routing support, including how to set up routing and forwarding, and how to configure the Gateway Routing Daemon (GateD).

Multiple Gateway Support

All hosts and gateways on a network store routing information, usually including a list of default gateway addresses.

The TCPware routing table contains a list of default gateway addresses. TCPware always uses the first gateway address on the list, unless it is marked as possibly being down. In this case, TCPware rotates the address of the gateway that is possibly down to the end of the list. TCPware then uses the next gateway address in the list, regardless of its state.

If all gateways are marked as being possibly down, TCPware uses all the addresses in rotation. This minimizes the number of datagrams sent to suspicious gateways, and maintains stability when more than one gateway is available.

Router or Link Failure

When a router fails, the host detects that it is sending packets into a "black hole." The host detects this in approximately one minute. The host:

1	Marks that entry in the gateway address list as possibly being down.
2	Rotates that gateway address entry to the end of the list.
3	Uses the next gateway address, which is now the first entry in the list.

When a link fails, the router connected to that link redirects TCPware to use another router for that destination. TCPware does this using ICMP redirects.

Router or Link Recovery

When a router recovers, TCPware reverts back to the router *only* if told to do so through a redirect for a specific destination. The acting router issues the redirect only if the original route has a better bandwidth, delay, and hop metric for the intended destination.

The system does not issue a redirect if the links between both routing paths are the same speed. In this case, TCPware continues to use the new router until:

- You reenter the gateway address using the Network Control Utility (NETCU)
- The new router fails

When a link recovers, TCPware discards the dynamic route set by the ICMP redirect and switches back to the original router.

Static Routing

This section explains how to configure specific routes using Network Control Utility (NETCU) commands.

Routing Guidelines

When setting up routing, consider the following guidelines:

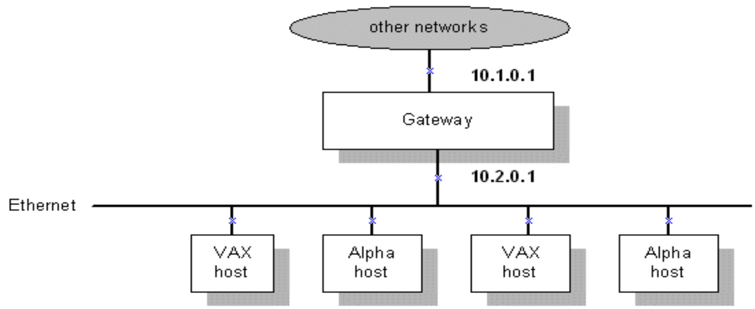
- Most routes should be network routes rather than host routes. This prevents the routing table from becoming too large.
- Define a default gateway using the NETCU SET GATEWAY command (see the *NETCU Command Reference*). Use the default gateway when sending a datagram to a host that is not on a local network and for which no other route is known.
- You can set up routing so that TCPware executes your routing commands at startup. Enter the NETCU routing commands in the TCPWARE:ROUTING.COM file. CNFNET creates this file during network configuration (see the following sections).
- If using GateD to configure routes, use GateD exclusively. Do not combine GateD routing with static routing set up in NETCU, as with ADD ROUTE. Route settings in the GATED.CONF file may conflict with settings in the static ROUTING.COM file.

Example 1

Figure 8-1 shows a local network connected to an internet through a gateway. Each VAX host runs TCPware for OpenVMS.

The gateway has an internet address for each network to which it connects.

Figure 8-1 Defining a Default Router



The easiest way to set up routing in this case is to define the gateway as the default gateway. To do this, perform one of the following tasks:

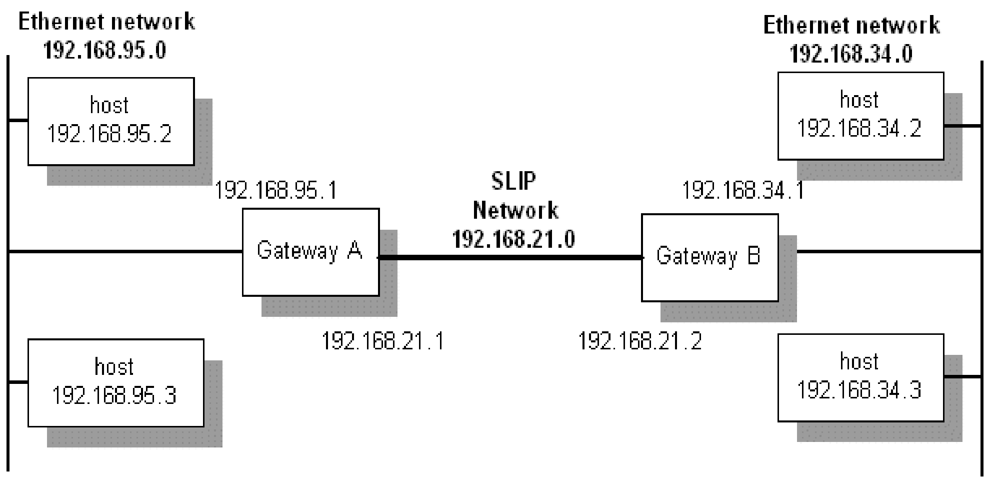
- Define the default gateway at each host by responding to prompts during TCPware's network configuration procedure (CNFNET).
- Enter the following NETCU command at the DCL prompt on each host:

  `NETCU SET GATEWAY 10.2.0.1`

Example 2

Figure 8-2 shows a sample internet consisting of three networks: Ethernet network 192.168.95.0, SLIP network 192.168.21.0, and Ethernet network 192.168.34.0.

Figure 8-2 Sample Internet



Each gateway has an internet address for each network to which it connects. This is how the networks are set up:

- At each TCPware host in network 192.168.95.0, set the local gateway host address:
  `SET GATEWAY 192.168.95.1`
- At each TCPware host on network 192.168.34.0, set the local gateway host address:
  `SET GATEWAY 192.168.34.1`
- At Gateway A, add the route through Gateway B's SLIP network address:
  `ADD ROUTE 192.168.34.0 192.168.21.2 /NETWORK /GATEWAY`
  `ENABLE FORWARDING`
- At Gateway B, add the route through Gateway A's SLIP address:
  `ADD ROUTE 192.168.95.0 192.168.21.1 /NETWORK /GATEWAY`
  `ENABLE FORWARDING`

You can also define the default gateway by responding to prompts during the network configuration procedure (CNFNET). See Chapter 3, *Configuring the TCP/IP Core Environment*, in the *Installation & Configuration Guide*.

Forwarding

Forwarding, if enabled using NETCU ENABLE FORWARDING, allows IPDRIVER to route (forward) datagrams between the available networks as needed.

IPDRIVER routes datagrams between networks when you enable forwarding, and there is a known route to the datagram's destination internet address. TCPware allows fragmentation of the routed datagram.

IPDRIVER transmits an Internet Control Message Protocol (ICMP) redirect message to the source internet address of the datagram if it routes the datagram over the same source network interface.

If you enable forwarding and ARP mode, TCPware responds to ARP requests for any nonlocal internet address for which it has a defined route. This is proxy ARP. The following example shows enabling forwarding in ARP mode:

```
NETCU ENABLE FORWARDING/ARP
```

TCPware does not forward multicast datagrams.

Multicast Routing

When an application wants to send datagrams to a multicast internet address (Class D, 224.0.0.0 through 239.255.255.255) and the application does not specify a multicast interface, TCPware determines the interface as follows:

1	If the routing table has a host route for the multicast address, TCPware uses the host route.

2	If the routing table has a default multicast route (a network route for 224.0.0.0), TCPware uses the default multicast route.
3	If the routing table has a default route, TCPware uses the default route.
4	Otherwise, TCPware uses the first multicast-capable interface it finds.

Using GateD

The Gateway Routing Daemon (GateD) manages multiple routing protocols, including the Routing Information Protocol (RIP), Local Network Protocol (HELLO), Router Discovery Protocol, Open Shortest Path First (OSPF) protocol, Exterior Gateway Protocol (EGP), and Border Gateway Protocol (BGP).

Using GateD, the network administrator can control the flow of routing information through a configuration language. Once you start GateD, it makes routing decisions based on the data gathered by the routing protocols. If routing using GateD, use GateD exclusively.

Note! If you want the system to function as a gateway, you must enable forwarding for it (using the ENABLE FORWARDING command in NETCU).

GateD allows you to control importing and exporting routing information by:

- Individual protocol
- Source and destination Autonomous System (AS)
- Source and destination interface
- Previous hop router
- Specific destination address

You can assign preference levels for different combinations of imported routing information by using a flexible masking capability. In TCPware, the name of the GateD process is TCPware_GateD.

GateD Configuration File

TCPware stores GateD configuration information in the TCPWARE:GATED.CONF file. You must create this file before you can use GateD.

For details on GateD configuration, see *GateD Configuration Statements*.

GateD Route Selection

GateD determines the "best" route using preference values set for each protocol or peer. Each route has a single associated preference value, even though you can set preferences at many places in the GATED.CONF file. The last (or most specific) preference value is the one GateD uses. Some protocols have a secondary preference, sometimes called a "tie-breaker."

The factors GateD uses in determining "best" routes include:

- The route with the numerically smallest **`preference`** value is preferred.
- For two routes with equal preferences, the route with the numerically smallest **`preference2`** (the "tie-breaker") is preferred.
- A route learned from an interior gateway protocol is preferred over a route learned from an exterior gateway protocol. Least preferred is a route learned indirectly by an interior protocol from an exterior protocol.
- If Autonomous System (AS) path information is available, it helps determine the most preferred route:
  - A route with an AS path is preferred over one without an AS path.
  - If the AS paths and origins are identical, the route with the lower metric is preferred.
  - A route with an AS path origin of interior protocol is preferred over one with an origin of exterior protocol. 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 lower metric is preferred.
- The route with the lowest numeric next-hop address is used.

Preference values range from 0 to 255. Table 8-1 summarizes the default preference values for routes learned in various ways.

Table 8-1 Default Routing Preference Values Defined by GateD Statements

Default `preference` value	**Is defined by ... statement**
0	interface
10	ospf
20	`gendefault` (internally generated default)
30	redirect
40	`kernel` (routes learned using the socket route)
60	static
90	hello
100	rip
110	(point-to-point interfaces)
120	`interfaces` (routes to interfaces that are down)
130	aggregate/generate
150	`ospf` (AS external)
170	bgp

Table 8-1 Default Routing Preference Values Defined by GateD Statements (Continued)

200	egp

Starting and Stopping GateD

After creating the TCPWARE:GATED.CONF file, you need to stop and restart GateD. Follow these steps:

1	Log in as the system manager.
2	Stop the GateD process by entering: **`@TCPWARE:SHUTNET GATED`**
3	Restart the GateD process by entering: **`@TCPWARE:STARTNET GATED`**

See the *Installation & Configuration Guide*, Chapter 6, *Starting and Testing TCPware*, for details on the STARTNET.COM and SHUTNET.COM command procedures.

GateD NETCU Commands

Use the NETCU commands in Table 8-2 to manage the GateD process. To use these commands, you need OPER or SYSPRV privilege.

See the *NETCU Command Reference*, Chapter 2, *NETCU Commands*.

Table 8-2 NETCU GateD Commands

Command	**Description**
CHECK GATED CONFIG	Checks a GateD configuration file for syntax errors
DUMP GATED STATE	Dumps the state of the GateD process to a file
LOAD GATED CONFIG	Loads a GateD configuration file
SET GATED TRACE	Controls tracing in GateD
SHOW GATED TRACE	Shows tracing in GateD
SHOW OSPF ADVERTISE	Shows OSPF link state advertisements
SHOW OSPF AS	Shows the AS external database entries
SHOW OSPF DESTINATIONS	Shows the list of destinations and their indices
SHOW OSPF ERRORS	Shows the OSPF error log

Table 8-2 NETCU GateD Commands (Continued)

Command	Description
SHOW OSPF HOPS	Shows the set of next hops for the OSPF router queried
SHOW OSPF INTERFACES	Shows all configured interfaces for OSPF
SHOW OSPF LOG	Shows the cumulative OSPF log of input/output statistics
SHOW OSPF NEIGHBORS	Shows all OSPF routing neighbors
SHOW OSPF ROUTING	Shows the OSPF routing table
SHOW OSPF STATE	Shows the link state database (except AS Externals)
SHOW RIP	Queries Routing Information Protocol (RIP) gateways
STOP/GATED	Stops the GateD process
TOGGLE GATED TRACING	Toggles tracing in GateD
UPDATE GATED INTERFACES	Rescans the GateD network interfaces

GateD Configuration Statements

The GateD configuration file is GATED.CONF. This file must be present for the GateD process to run. The structure of the GateD configuration language is similar to C. The configuration file consists of statements terminated by a semicolon (;). Statements consist of tokens separated by a space. This structure simplifies identification of the associated parts of the configuration.

You can include comment lines either by beginning them with a pound sign **(#)** or delimiting them with slash asterisk **(/*)** and asterisk slash **(*/)**. The configuration file consists of the following sections, which reflect the order in which the statements, if used, must appear:

Directives (`%directory`, `%include`)	`routerdiscovery server`	`redirect`	`hello`	`rip`
Route Filtering (`network` and `mask` statements)	`routerdiscovery client`	`generate`	`import`	`bgp`
Definition Statements (`autonomoussytem`, `routerid`, `martians`)	`traceoptions`	`static`	`export`	`egp`
`interfaces`	`aggregate`	`options`	`icmp`	`ospf`

Directives

Directive statements include:

- **%directory**
- **%include**

Directive statements provide special instructions to the parser. They do not relate to the protocol configuration and can occur anywhere in GATED.CONF. They also end in a new line instead of a semicolon (;) like the other statements.

Format

%directory *"directory"*

Defines the directory where the include files go if you do not fully specify directory as part of the filename in the %include statement. Does not actually change the current directory, but simply applies the directory prefix.

%include *"filename"*

Identifies an include file. GateD includes the contents of the file in GATED.CONF at the point where the %include appears. If you do not fully specify the filename, it is relative to the directory defined in %directory. The %include directive causes GateD to parse the specified file completely before resuming. You can nest up to ten levels of include files.

Table 8-3 Global Trace Options

Option	Description
`adv`	For debugging: traces the allocation and freeing of policy blocks.
`all`	Turns on the `general, normal, policy, route, state, task,` and `timer` options.
`general`	Shorthand for specifying both the `normal` and `route` options.
`iflist`	Traces reading of the kernel interface. Useful to specify this with the `-t` option on the command line since the first interface scan occurs before reading the configuration file.
`normal`	Traces normal protocol occurrences (abnormal protocol occurrences are always traced).
`parse`	For debugging: traces the lexical analyzer and parser.
`policy`	Traces how protocol and user-specified policy apply to routes imported and exported.
`route`	Traces routing table changes for routes installed by the protocol or peer.
`state`	Traces state machine transitions in the protocols.

Table 8-3 Global Trace Options (Continued)

Option	Description
`symbols`	Traces symbols read from the kernel at startup. The only useful way to specify this level of tracing is to use the `-t` option on the command line, since GateD reads the symbols from the kernel before parsing the configuration file.
`task`	Traces system interface and processing associated with the protocol or peer.
`timer`	Traces timer usage by the protocol or peer.

traceoptions

The **`traceoptions`** statement controls tracing options. You can configure GateD tracing options at many levels. These include file specifications, control options, and global and protocol-specific tracing options.

Lower levels of statements inherit tracing options from the next higher level, unless overridden.

Format

traceoptions ***["tracefile" [replace] [size size[*****k | m*****] files files]]***
***[nostamp] traceoptions [except traceoptions]* | none ;**

Options and Parameters

"tracefile"

File to receive tracing information. If this filename is not fully specified, GateD creates it in the directory where you started GateD.

replace

Replaces an existing file. The default is to append to an existing file.

size *size[*k | m*]* files *files*

Limits the maximum size, in k or m or the files indicated, of the trace file (the minimum is 10k). When the file reaches size, GateD creates a new version.

nostamp

Control option which means not to prepend a timestamp to all trace lines. The default is to prepend a timestamp.

traceoptions

Specific to each protocol statement. The global trace options appear in Table 8-3. Note that these global options may not apply to all protocols.

except *traceoptions*

Disables more specific trace options after enabling broader ones.

none

Turns off all tracing for the protocol or peer.

options

The options statements let you specify some global options. If used, options must appear before any other type of configuration statement in GATED.CONF.

Format

options *[*nosend*]*
**   *[*noresolv*]***
**   *[*gendefault *[*preference *value][gateway host]]***
**   *[*syslog *[*upto*] loglevel]***
**   *[*mark *time]* ;**

Options and Parameters

nosend

Does not send packets. Makes it possible to run GateD on a live network to test protocol interactions, without actually participating in the routing protocols. You can examine the packet traces in the GateD log to verify that GateD functions properly. Most useful for RIP and HELLO. Does not yet apply to BGP, and not useful with EGP and OSPF.

noresolv

Does not resolve symbolic names into IP addresses. By default, GateD uses the `gethostbyname()` and `getnetbyname()` library calls that 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. Use this option to prevent these calls.

Note! When you use this option, symbolic names cause configuration file errors.

gendefault *[*preference *value] [*gateway *host]*
nogendefault

Creates a default route with the special protocol default when a BGP or EGP neighbor is up. You can disable this for each BGP/EGP group with the **`nogendefault`** option. By default, this route has a **`preference`** value of **`20`**. This route is normally not installed in the kernel forwarding table; it is only present for announcement to other protocols. The **`gateway`** option installs the default route in the kernel forwarding table with a next hop of the gateway defined.

Note! Using more general options is preferred to using **`gendefault`**. (See *Aggregate* for details on the **`generate`** statement.)

syslog *[*upto*] loglevel*

Amount of data GateD logs to OPCOM. OpenVMS systems map UNIX `syslog` logging levels to OPCOM severity levels. The default is **`syslog upto info`**. The mapping of `syslog` to OPCOM logging levels appears in Table 8-4.

mark *time*

GateD sends a message to the trace log at the specified *time* interval. Can be one method of determining if GateD is still running.

Table 8-4 Mapping of UNIX syslog Levels to OpenVMS OPCOM Severity Levels

syslog log level	Is equivalent to OPCOM level...
`emerg`	FATAL
`alert`	FATAL
`crit`	FATAL
`err`	ERROR
`warning`	WARNING
`notice`	INFORMATIONAL
`info` (default)	INFORMATIONAL
`debug`	INFORMATIONAL

Example

```
#  generate a default route when peering with an EGP or BGP neighbor:
#
options gendefault ;
```

interfaces

An interface is the connection between a router and one of its attached networks. Specify a physical interface by interface name, IP address, or domain name. Multiple reference levels in the configuration language let you identify interfaces using wildcards (only the device driver part of the name, to match any unit number), interface type names, or addresses.

Format

interfaces {
options
[*strictinterfaces*]
***[*scaninterval *time]* ;**
interface *list*
[*preference value*]
[*down preference *value]
[*passive*]
[*simplex*]
[*reject*]
***[*blackhole*]* ;**
define *address*
[*broadcast *address]* | *[*pointtopoint *address]
[*netmask *mask]
***[*multicast*]* ;**
} ;

Options Clause

options
[*strictinterfaces*]
***[*scaninterval *time]* ;**

strictinterfaces

Makes it a fatal error to use reference interfaces not present when you start GateD or that are not part of the **`define`** parameter. Normally, GateD issues a warning message and continues.

scaninterval *time*

Sets 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 (such as BSD 4.4).

Interface Clause

Sets interface options on the specified interfaces. A *list* can consist of interface names, domain names, numeric addresses, or the value **`all`**. Include one or more interface names, including wildcard names (without a number) and those that can specify more than one interface or address.

There are three ways to reference an interface:

By wildcard	Only the device driver part of the name, to match any unit number.
By name	Combined device driver and unit number of an interface.
By address	IP address or domain name (if resolving to one address only).

There are four types of interfaces allowed:

Loopback	Must have the address 127.0.0.1. Packets from a loopback interface go back to the originator. Also used for reject and blackhole routes (**not supported in TCPware**). The interface ignores any net mask. It is useful to assign an additional address to the loopback interface that is the same as the OSPF or BGP router ID; this allows routing to a system based on router ID that works if some interfaces are down.
Broadcast	Multiaccess interface capable of physical level broadcast, such as Ethernet, Token-Ring, and FDDI. A broadcast 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	Tunnel to another host, usually on some sort of serial link. A point-to-point interface has a local address and a remote address. The remote address must be unique among the interface addresses on a given router. Many point-to-point interfaces and up to one non point-to-point interface must share the local address. This conserves subnets as you do not need any when using this technique. If you use a subnet mask on a point-to-point interface, only RIP version 1 and HELLO use it to determine which subnets propagate to the router on the other side of the point-to-point interface.
Nonbroadcast multiaccess (NBMA)	Multiaccess but not capable of broadcast, such as frame relay and X.25. This type of interface has a local address and a subnet mask.

preference *value*

Sets the preference for routes to this interface when it is up and GateD determines it to function properly. The default preference *value* is **`0`**. While the `preference` statement is optional, it is strongly recommended that you set an explicit preference value if you do use it.

down preference *value*

Sets the preference for routes to this interface when GateD determines that it does not function properly, but the kernel does not indicate that it is down. The default down preference *value* is **`120`**.

passive

Does not change the preference of the route to the interface if determined not to function properly from lack of routing information. GateD checks this only if the interface actively participates in a routing protocol.

simplex

The interface does not recognize its own broadcast packets. Some systems define an interface as simplex with the IFF_SIMPLEX flag. On others, the configuration defines it. On simplex interfaces, packets from the local host are assumed to have been looped back in software and are not used to indicate that the interface functions properly.

reject, blackhole

Not supported in TCPware. Normally, this uses the address of the interface that matches these criteria as the local address when installing reject routes in the kernel. A blackhole route is like a reject route except that it does not support **`unreachable`** messages.

Define Clause

interfaces {
.
.
.
define ***address***
[**broadcast** ***address] | [*****pointtopoint** ***address]***
[**netmask** ***mask]***
[**multicast*****]*** **;**
} ;

Defines interfaces not present when starting GateD so that the configuration file can reference them when using `options strictinterfaces`.

broadcast ***address***

Makes the interface broadcast-capable (for Ethernet or Token-Ring) and specifies the broadcast address.

pointtopoint ***address***

Makes the interface point-to-point (such as SLIP or PPP) and specifies the address on the local side of the interface. The first address in the **`define`** statement references the host on the remote end of the interface.

An interface not defined as **`broadcast`** or **`pointtopoint`** must be nonbroadcast multiaccess (NBMA), such as for an X.25 network.

netmask ***mask***

Subnet mask to use on the interface. Ignored on point-to-point interfaces.

multicast

Makes the interface multicast-capable.

Examples

1 This example sets the interface as passive.

```
#  do not mark interface 192.168.95.31 as down,
#  even if there is no traffic:
#
interfaces{
  interface 192.168.95.31 passive ;
} ;
```

2 This example shows the interface statements used with the **`rip`** statement (see the **`rip`** description). Users would receive RIP packets only from interfaces `sva-0` and `sva-1`, but not from `fza-0`, and `sva-1` would be the only one that could send them.

```
rip yes {
  interface all noripin noripout ;
  interface sva ripin
;
  interface sva-1 ripout ;
  } ;
```

Definition Statements

Definition statements include:

- autonomoussystem
- routerid
- martians

Definition statements are general configuration statements that relate to all of GateD or at least to more than one protocol. You must use these statements for any protocol statements in the configuration file.

Format

autonomoussystem *ASnumber [*loops *number]* ;

An autonomous system (AS) is a set of routers under a single technical administration, using an internal protocol and common metrics to route packets within the AS, and an external protocol to route packets to other ASs. The Network Information Center (NIC) assigns AS numbers.

The **`autonomoussystem`** statement sets the AS number of the router. You require this option if using BGP or EGP. The **`loops`** option is only for protocols supporting AS paths, such as BGP. It controls the number of times this AS can appear in an AS path, and defaults to **`1`**.

routerid *host* ;

A router ID is an IP address used as a unique identifier assigned to represent a specific router, usually the address of an attached interface. The **`routerid`** statement sets the router ID for the BGP and OSPF protocols. The default is the address of the first interface GateD encounters. 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.

martians {
host *host [*allow*]* ;
network *[*allow*]* ;
***network* mask *mask [*allow*]* ;**
***network* masklen *number [*allow*]* ;**
default *[*allow*]* ;
} ;

The `martians` statement defines a list of invalid addresses, called *martians*, that the routing software ignores. Sometimes a misconfigured system sends out obviously invalid destination addresses. The statement allows additions to the list of martian addresses. (See *Route Filtering* for details on specifying ranges.)

You can also use the **`allow`** parameter to explicitly allow a subset of an otherwise disallowed range.

Example

This example shows the use of all three definition statements, **`autonomoussystem, routerid,`** and **`martians`**.

```
#  use AS number 249:
#
autonomoussystem 249 ;
#
#  set the router
ID number:
#
routerid 192.168.95.31 ;
#
#  prevent routes to
0.0.0.26 from ever being accepted:
#
martians {
host 0.0.0.26 ;
};
```

Route Filtering

You can filter routes by matching a certain set of routes by destination, or by destination and mask. Use route filters on **`martians, import,`** and **`export`** statements.

The action taken when no match is found depends on the context. For example, import and export route filters assume an **`all reject ;`** at the end of a list. A route matches the most specific filter that applies. Specifying more than one filter with the same destination, mask, and modifiers generates an error.

Format

network [*exact | refines | allow*]
network* mask *mask [*exact | refines*]
network* masklen *number [*exact | refines*]
all
default
host *host*

Options and Parameters

network

Destination network IP address. You can use one of the following options:

`exact`	Destination mask must match the supplied mask exactly. Used to match a network, but no subnets or hosts of that network.
`refines`	Destination mask must be more specified (longer) than the filter mask. Used to match subnets or hosts of a network, but not the network.
`allow`	See the **martians** definition statement.

mask *mask*

Destination network mask.

masklen *number*

Length of the destination network mask.

all

Entry matches anything. 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. Equivalent to **`0.0.0.0 mask 0.0.0.0 exact.`** (Not valid for **`martians`** statements.)

host *host*

Matches the specific host. To match, the address must match exactly the specified host, and the network mask must be a host mask (all 1s). Equivalent to **`host mask 255.255.255 exact.`** (Not valid for **`martians`** statements.)

rip

GateD supports the Routing Information Protocol (RIP). RIP is a distance-vector protocol for distributing routing information at the local network level of the Internet. In distance-vector routing, each router transmits destination addresses and costs to its neighbors (computers communicating over RIP).

RIP versions 1 and 2 are 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 uses the fewest gateways, that is, the shortest path, not taking into account congestion or delay along the way.

RIP uses two types of packets: requests and responses.

Requests. A request asks for information about specific destinations or for all destinations. RIP can send requests when a router:

- Comes up
- Receives timed-out information about a destination

If a request fails to specify a destination, RIP assumes the router requests information about all destinations.

Responses. Responses contain destination and cost pairs. RIP sends responses under the following three conditions:

- In response to a request
- When information changes; for example, cost information
- At set intervals; for example, reporting the destination to each neighbor every 30 seconds

RIP discards the destination and cost information if a neighbor fails to report the distance to a destination after a certain time interval.

RIP IP Addresses. RIP version 1 contains no provision for passing around a mask. RIP infers the mask based on whether the address is class A, B, or C. Sometimes there are special cases when the inferred mask differs from class A, B, or C. For example:

- When you use RIP with a subnet (in this case the routers must know the subnet mask for a particular network number)
- When the system updates RIP with an address reported as 0.0.0.0, RIP considers this address as a default destination with a mask of 0.0.0.0
- When the system updates RIP with bits set in the host portion of the address, RIP assumes the address refers to a host with a mask of 255.255.255.255

With RIP version 2, you can specify the network mask with each network in a packet.

Configuring RIP. You configure RIP in the GATED.CONF file using a GateD protocol statement that enables or disables RIP. The syntax of the **`rip`** statement is as follows, with the parameters described next:

Format

rip yes | no | on | off
[{[no]broadcast ;
nocheckzero ;
preference *value* **;**
defaultmetric *metric* **;**
query authentication *[* **none |** *[[***simple | md5***]* **password** *]]* **;**
interface *list*
*[[***no***]***ripin** *] [[***no***]***ripout** *]*
*[***metricin** *metric]*
*[***metricout** *metric]* **;**
*[***version 1***]* **|** *[* **version 2** *[***multicast | broadcast***]]*
*[[***secondary***]* **authentication** *[* **none |** *[[***simple | md5***] password]]]* **;**

trustedgateways *list* **;**

sourcegateways *list* **;**

traceoptions *options* **;**

}*]* **;**

Options and Parameters

yes | on (default)
no | off

When enabled on a host, RIP listens in the background to routing updates. When enabled on a gateway, RIP supplies routing updates. Enabled by default.

broadcast ;

Broadcasts RIP packets regardless of the number of interfaces present. Useful when propagating static routes or routes learned from another protocol into RIP. In some cases, using **`broadcast`** when only one network interface is present can cause data packets to traverse a single network twice. The default for more than one interface.

nobroadcast ;

Does not broadcast RIP packets on attached interfaces even if there is more than one. If you use the **`sourcegateways`** parameter, routes are still unicast directly to that gateway. The default for a single interface.

nocheckzero ;

Does not make sure that reserved fields in incoming RIP version 1 packets are zero. Normally RIP rejects packets whose reserved fields are zero.

preference *value* **;**

Sets the preference for routes learned from RIP. A preference specified in import policy can override this. The default preference *value* is **`100`**.

defaultmetric *metric* ;

Metric used when advertising routes learned from other protocols. Choice of values requires that you explicitly specify a metric in order to export routes from other protocols into RIP. A metric specified in export policy can override this. The default *metric* is **`16`**.

query authentication ;

Authentication required of query packets that do not originate from routers. The default is **none**.

Interface Clause

rip yes | no | on | off
*[{[***no***]***broadcast ;**
nocheckzero ;
preference *value* **;**
defaultmetric *metric* **;**
query authentication *[* **none |** *[[***simple | md5***] password]]* **;**
interface *list*
*[[***no***]***ripin** *] [[***no**]**ripout** *]*
*[***metricin** *metric]*
*[***metricout** *metric]* **;**
*[***version 1***]* **|** *[* **version 2** *[***multicast | broadcast***]]*
*[[***secondary***]* **authentication** *[***none |** *[[***simple | md5***]* **password***]]* **;**
trustedgateways *list* **;**
sourcegateways *list* **;**
traceoptions *options* **;**
}] ;

Controls various attributes of sending RIP on specific interfaces. (See the interfaces statement for a description of *list*.) Note that if there are multiple interfaces configured on the same subnet, only the first one on which RIP output is configured sends the RIP updates. This limitation is required because of the way the UNIX kernel operates. A future GateD release will hopefully remove this limitation. The default *list* value is `all.`

ripin (default)
noripin

Use **`ripin`** explicitly when using **`noripin`** on a wildcard interface descriptor. The **`noripin`** option ignores RIP packets received over the specified interfaces.

ripout (default)
noripout

Use **`ripin`** explicitly when using **`noripout`** on a wildcard interface descriptor. The **`noripin`** does not send RIP packets over the specified interfaces.

metricin ***metric***

RIP metric to add to incoming routes before they are installed in the routing table. Makes the router prefer RIP routes learned using the specified interfaces less than those learned from other interfaces. The default is the kernel interface metric plus 1. If using this as the absolute value, the kernel metric is not added.

metricout ***metric***

RIP metric to add to routes sent over the specified interface(s). Makes other routers prefer other sources of RIP routes over this router. The default *metric* value is **`0`**.

version 1 (default)

Sends RIP version 1 packets over the specified interface(s).

version 2 *[*multicast | broadcast*]*

Sends RIP version 2 packets over the specified interfaces. If IP multicasting support is available on this interface, the default is to send full version 2 packets. If multicasting is not available, version 1 compatible version 2 packets are sent. Options include:

`multicast` (default)	Multicasts RIP version 2 packets over this interface.
`broadcast`	Broadcasts RIP version 1 compatible version 2 packets over this interface even if IP multicasting is available

[*secondary*]* authentication *[*none | *[[*simple | md5*]* password*]]

Authentication type to use. Applies only to RIP version 2 and is ignored for RIP-1 packets. If you specify a *password*, the authentication type defaults to **`simple`**. The password should be a quoted string with 0 to 16 characters. If you specify **`secondary`**, this defines the secondary authentication. The default is **`authentication none`**.

Remaining Options and Parameters

rip yes | no | on | off
[{[no]broadcast ;
nocheckzero ;
preference *value* ;
defaultmetric *metric* ;
query authentication *[* none | *[[*simple | md5*]* password *]]* ;
interface *list*
[[*no]ripin *] [[*no*]*ripout *]
[*metricin *metric]
***[*metricout *metric]* ;**
[version 1*]* | *[* version 2 *[*multicast | broadcast*]]*
***[[*secondary*]* authentication *[*none | *[[*simple | md5*]* password *]]*;**
trustedgateways *list* ;
sourcegateways *list* ;
traceoptions *options* ;
}*]* ;

trustedgateways *list*

List of gateways from which RIP accepts updates (host names or IP addresses). If used, only updates from the gateways in the list are accepted. The default *list* value is **`all`**.

sourcegateways *list*

List of routers to which RIP sends packets directly, not through multicasting or broadcasting. If used, only updates from the gateways in the list are accepted. The default *list* value is **`all`**.

traceoptions *options*

RIP-specific trace options:

packets	All RIP packets, or packets *[***detail***]* **send** or *[***detail***]* **recv** (**detail** provides a more verbose format to provide more details; if used, **detail** must come before **send** or **recv**)
request	RIP information request packets, such as REQUEST, POLL and POLLENTRY
response	RIP RESPONSE packets that actually contain routing information

hello

GateD supports the HELLO protocol. HELLO is an interior protocol that uses delay as the deciding factor when selecting the best route. Delay is the round trip time between source and destination. HELLO is not as widely used as when it was the interior protocol of the original 56-Kb/sec NSFNET backbone and used between LSI-11 ("fuzzball") routers. Because of this, HELLO is disabled by default.

By default, HELLO, like RIP, uses the kernel interface metric set by the **`ifconfig`** command to influence metrics added to routes as they are installed in the routing table (**`metricin`**). Since the kernel interface metric is in hops, it must be translated into HELLO's millisecond metric. For the translation scheme, see Table 8-5.

Table 8-5 HELLO Hops-to-Metrics Translation

This many Hops	Translate to this HELLO metric	This many Hops	Translate to this HELLO metric	This many Hops	Translate to this HELLO metric
0	0	6	713	12	75522
1	100	7	1057	13	11190
2	148	8	1567	14	16579
3	219	9	2322	15	24564
4	325	10	3440	16	3000
5	481	11	5097		

You configure HELLO in the GATED.CONF file using a GateD protocol statement that enables or disables HELLO.

When enabled, HELLO assumes **`nobroadcast`** when only one interface exists. HELLO assumes broadcast when more than one interface exists.

Format

hello yes | no | on | off
    ***[{[no]*broadcast ;**
      preference *value* ;
      defaultmetric *metric* ;
      interface *list*
        [[no]*helloin *]
        [[no]*helloout *]
        [*metricin *metric]
       ***[*metricout *metric]* ;**
     trustedgateways *list* ;
     sourcegateways *list* ;

traceoptions ***options*** **;**
}*]*
;

Options and Parameters

yes | on or no | off (default)

When enabled on a host, HELLO listens in the background for routing updates. When enabled on a gateway, HELLO supplies routing updates. Disabled by default.

broadcast ;
nobroadcast ;

The **`broadcast`** option broadcasts HELLO packets regardless of the number of interfaces present. Useful when propagating static routes or routes learned from another protocol into HELLO. In some cases, using **`broadcast`** when only one network interface is present can cause data packets to traverse a single network twice. The default for more than one interface.

The **`nobroadcast`** option does not broadcast HELLO packets on attached interfaces, even if there is more than one. If you use the **`sourcegateways`** parameter, routes are still unicast directly to that gateway. The default for a single interface.

preference ***value*** **;**

Preference for routes learned from HELLO. A preference specified in import policy can override this. The default preference *value* is **`90`**.

defaultmetric ***metric*** **;**

Metric used when advertising routes learned from other protocols. Requires you to explicitly specify a metric in order to export routes from other protocols into HELLO. A metric specified in export policy can override this. The default *metric* is **`30000`**.

Interface Clause

interface ***list***
[[*no*]*helloin *]
[[*no*]*helloout *]
[*metricin *metric]
[*metricout *metric] **;**

Controls various attributes of sending HELLO on specific interfaces. (See interfaces statement for a description of *list*.) Note that if there are multiple interfaces configured on the same subnet, only the first interface that has HELLO output configured sends the HELLO updates. This limitation is required because of the way the UNIX kernel operates. A future GateD release will hopefully remove this limitation. The default interface *list* value is `all`.

helloin (default)
nohelloin

Use **`helloin`** explicitly when using **`nohelloin`** on a wildcard interface descriptor. The

`nohelloin` option ignores HELLO packets received over the specified interfaces.

hellout (default)
nohelloout

Use **`helloout`** explicitly when using **`nohelloout`** on a wildcard interface descriptor. The **`nohelloout`** option does not send HELLO packets over the specified interfaces.

metricin *metric*

HELLO metric to add to incoming routes before GateD installs them in the routing table. Makes this router prefer HELLO routes learned from other interfaces over those from the specified interface(s). The default is the kernel interface metric plus one. If using this as the absolute value, GateD does not add the kernel metric to the routing table.

metricout *metric*

HELLO metric to add to routes that are sent over the specified interface(s). Makes other routers prefer other sources of HELLO routes over this router. The default metric out *metric* value is **`0`**.

Remaining Options and Parameters

hello yes | no | on | off
[{ *[*no*]*broadcast ;
preference *value* ;
defaultmetric *metric* ;
interface *list*
*[[*no*]*helloin *]*
*[[*no]helloout *]*
*[*metricin *metric]*
*[*metricout *metric]* ;
trustedgateways *list* ;
sourcegateways *list* ;
traceoptions *options* ;
}*]* ;

trustedgateways *list*

List of gateways from which HELLO accepts updates (host names or IP addresses). If used, HELLO accepts only updates from the gateways in the list. The default *list* value is **`all`**.

sourcegateways *list*

List of routers to which HELLO sends packets directly, not through multicasting or broadcasting. If used, HELLO accepts only updates from the gateways in the list. The default *list* value is **`all`**.

traceoptions packets

All HELLO packets, or packets *[*`detail`*]* **`send`** or *[*`detail`*]* **`recv`** (**`detail`** provides a more verbose format to provide more details; if used, **`detail`** must come before **`send`** or **`recv`**).

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.

Currently the only reason to specify the **`icmp`** statement is to be able to trace the ICMP messages that GateD receives.

Format

icmp { traceoptions *options* ; }

Options and Parameters

traceoptions *options* ;

ICMP tracing options (which you can modify with **`detail`** and **`recv`**) are as follows:

`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 timestamp request/response
`error`	Only ICMP error packets, which include time exceeded, parameter problem, unreachable and source quench

redirect

GateD controls whether ICMP redirect messages can modify the kernel routing table. If disabled, GateD only prevents a system from listening to ICMP redirects. By default, ICMP redirects are enabled on hosts, and disabled on gateways that run as RIP or HELLO suppliers.

You configure ICMP redirect handling in the GATED.CONF file using a GateD protocol statement.

Format

redirect yes | no | on | off
  ***[*{preference *value* ;**
    interface *list [[*no*]*redirects *]* ;
    trustedgateways *list* ;
}*]*
;

Options and Parameters

yes | on
no | off

Enabled by default on hosts. Disabled by default on gateways running as RIP or HELLO suppliers.

preference *value*

Preference for routes learned from a redirect. The default preference *value* is **`30`**.

interface *list [[*no*]*redirects *]*

Enables and disables redirects interface by interface. (See **interfaces** for a description of *list*.) The default interface *list* value is **`all`**. The possible parameters are:

`redirects` `(default)`	May be necessary when you use **`noredirects`** on a wildcard interface descriptor.
`noredirects`	Ignores redirects received over the specified interface(s). The default is to accept redirects on all interfaces.

trustedgateways *list*

List of gateways from which redirects are accepted (host names or addresses). By default, all routers on the shared network(s) are trusted to supply redirects. If used, only redirects from the gateways in the list are accepted. The default *list* value is **`all`**.

routerdiscovery server

The Router Discovery Protocol is an IETF standard protocol used to inform hosts of the existence of routers without having hosts wiretap routing protocols such as RIP. Use it in place of, or in addition to, statically configured default routes in hosts.

The protocol is in two parts, the server that runs on routers and the client that runs on hosts (see the next statement). GateD treats these much like two separate protocols that you can enable only one at a time.

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 router 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 occurring 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 an Advertisement Lifetime field indicating how long the advertised addresses are valid. This lifetime is configured such that another Router Advertisement is sent before the lifetime expires. A lifetime of zero indicates that one or more addresses are no longer valid.

On systems supporting IP multicasting, the Router Advertisements are sent to the all-hosts multicast address 224.0.0.1 by default. However, you can specify **`broadcast`**. 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.

Note! Do not mix **`routerdiscovery server`** and **`routerdiscovery client`** statements in the GATED.CONF file or you may get unintended results. You should also include **`preference`** statements in the **`interfaces`** and **`routerdiscovery`** statements whenever possible.

Format

routerdiscovery server yes | no | on | off
 ***[*{ traceoptions state ;**
  interface *list*
   [*minadvinterval *time]
   [*maxadvinterval time*]
   ***[*lifetime time*]* ;**
  address *list*
   [*advertise*]* | *[*ignore*]
   [*broadcast*]* | *[*multicast*]
   ***[*ineligible*]* | *[*preference *value]* ;**
}*]* ;

Options and Parameters

yes | on
no | off

Enables or disables Router Discovery Protocol Server.

traceoptions state

The **`state`** is the only trace option, which traces the state transitions. The Router Discovery Server does not directly support packet tracing options; tracing of router discovery packets is enabled through the **`icmp`** statement described in the **icmp** statement section.

Interface Clause

routerdiscovery server yes | no | on | off
[{ traceoptions state ;
interface *list*
[*minadvinterval *time]
[*maxadvinterval time*]
***[*lifetime time*]* ;**
address *list*
[*advertise*]* | *[*ignore*]
[*broadcast*]* | *[*multicast*]
[ineligible*]* | *[*preference value*]* ;

interface *list*

Parameters that apply to physical interfaces. Note a slight difference in convention from the rest of GateD: **`interface`** specifies just physical interfaces, while **`address`** specifies protocol (in this case, IP) addresses.

maxadvinterval *time*

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). The default is **`10:00`** (10 minutes).

minadvinterval *time*

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`** X **`maxadvinterval`**.

lifetime *time*

Lifetime of addresses in a Router Advertisement. Must be no less than **`maxadvinterval`** and no greater than **`2:30:00`** (two hours, thirty minutes). The default is **`3`** X **`maxadvinterval.`**

Address Clause

routerdiscovery server yes | no | on | off
***[*{ traceoptions state ;**

interface ***list***
[minadvinterval *time]*
[maxadvinterval time]
[lifetime time] ;
address *list*
[advertise] | [ignore]
[broadcast] | [multicast]
[preference *value]* | [ineligible] ;

address *list*

Parameters that apply to the specified set of addresses on this physical interface. Note a slight difference in convention from the rest of GateD: **`interface`** specifies just physical interfaces while **`address`** is protocol (in this case, IP) addresses.

advertise (default)
ignore

The **`advertise`** keyword includes the specified addresses in Router Advertisements. The **`ignore`** keyword does not.

broadcast
multicast

The **`broadcast`** keyword includes the given addresses in a broadcast Router Advertisement because this system does not support IP multicasting, or some hosts on an 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. This is the default if the router does not support IP multicasting.

The **`multicast`** keyword includes the given addresses in a multicast Router Advertisement. If the system does not support IP multicasting, the address(es) is not 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 are included in a broadcast Router Advertisement.

preference *value*
ineligible

The **`preference`** keyword sets the preferability of the addresses as a default router address, relative to other router addresses on the same subnet. A 32-bit, signed, two's complement integer, with higher values meaning more preferable. Note that hex `80000000` may only be specified as ineligible. The default value is **`0`**. Use a **`preference`** statement whenever possible.

The **`ineligible`** keyword assigns the given addresses 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.

routerdiscovery client

A host listens for Router Advertisements through 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 a non-zero lifetime is received, the host installs a default route to each of the advertised addresses. If the preference is 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 happens when a Router Advertisement is not received to refresh these routes before the lifetime expires.

Note! Do not mix routerdiscovery server and routerdiscovery client statements in the GATED.CONF file or you may get unintended results.
You should also include preference statements in the interfaces and routerdiscovery statements whenever possible.

Format

routerdiscovery client yes | no | on | off
  traceoptions state ;
   preference *value* ;
   interface *list*
    [*enable*]* | *[*disable*]
    [*broadcast*]* | *[*multicast*]
    [quiet*]* | *[*solicit*]* ;
}*]* ;

Options and Parameters

yes | on
no | off

Enables or disables the Router Discovery Protocol Client.

traceoptions state ;

The **`state`** is the only trace option, which traces the state transitions. The Router Discovery Server does not directly support packet tracing options; tracing of router discovery packets is enabled through the **`icmp`** statement described in the **icmp** statement section.

preference *value* ;

Preference of all Router Discovery default routes. Use a preference statement whenever possible. Default is **`55`**.

Interface Clause

routerdiscovery client yes | no | on | off
{ traceoptions state ;
preference *value* ;
interface *list*
[*enable*]* | *[*disable*]
[*broadcast*]* | *[*multicast*]
***[*solicit*]* | *[*quiet*]* ;**
}*]* ;

interface *list*

Parameters that apply to physical interfaces. Note a slight difference in convention from the rest of GateD: **`interface`** specifies just physical interfaces. The Router Discovery Client has no parameters that apply only to interface addresses.

enable (default)
disable

Either performs or does not perform Router Discovery on the specified interfaces.

broadcast
multicast

The **`broadcast`** keyword broadcasts Router Solicitations on the specified interfaces. This is the default if IP multicast support is not available on this host or interface.

The **`multicast`** keyword multicasts Router Solicitations on the specified interfaces. If IP multicast is not available on this host and interface, no solicitation is performed. The default is to multicast Router Solicitations if the host and interface support it, otherwise Router Solicitations are broadcast.

solicit (default)
quiet

Either sends or does not send Router Solicitations on this interface, even though Router Discovery is performed.

egp

GateD supports the Exterior Gateway Protocol (EGP). EGP is an exterior routing protocol that moves routing information between Autonomous Systems (ASs). Unlike interior protocols, EGP propagates only reachability indications, not true metrics. EGP updates contain metrics, called distances, which range from 0 to 255. GateD only compares EGP distances learned from the same AS. EGP currently has limited usage. By default, EGP is disabled.

Before EGP sends routing information to a remote router, it must establish an adjacency with that router. This occurs by exchanging Hello and I Heard You (I-H-U) messages with that router. (Hello should not to be confused with the HELLO protocol, or OSPF HELLO messages.) Computers communicating over EGP are called EGP neighbors, and the exchange of Hello and I-H-U messages is known as acquiring a neighbor.

Once you acquire a neighbor, the system polls it for routing information. The neighbor responds by sending an update containing routing information. If the system receives a poll from its neighbor, it responds with its own update packet. When the system receives an update, it includes routes from the update into its routing database. If the neighbor fails to respond to three consecutive polls, GateD assumes that the neighbor is down and removes the neighbor's routes from its database.

You configure EGP in the GATED.CONF file using a GateD protocol statement.

Format

```
egp  yes | no | on | off
    [{  preference value ;
       defaultmetric metric ;
       packetsize max ;
       traceoptions options ;
       group
           [peeras ASnumber]
           [localas ASnumber]
           [maxup number
       {  neighbor host
               [metricout metric]
               [preference value]
               [preference2 value]
               [ttl ttl]
               [nogendefault]
               [importdefault]
               [exportdefault]
               [gateway gateway]
               [lcladdr local-address]
               [sourcenet network]
               [minhello | p1 time]
               [minpoll | p2  time]
               [traceoptions options] ;
     } ;
}] ;
```

Options and Parameters

yes | on
no | off (default)

Enables or disables EGP support. Disabled by default.

preference *value* ;

Preference for routes learned from EGP. A preference specified on the **`group`** or **`neighbor`** statements or by import policy can override this. The default preference *value* is **`200`**.

defaultmetric *metric* ;

Metric used when advertising routes over EGP. This choice of values requires you to explicitly specify a metric when exporting routes to EGP neighbors. A metric specified on the **`neighbor`** or **`group`** statements or in export policy can override this. The default *metric* is **`255`**.

packetsize *max* ;

Maximum size of a packet that EGP expects to receive from this neighbor. If EGP receives a larger packet, it is incomplete and EGP discards it. EGP notes the length of this packet and increases the expected size to be able to receive a packet of this size. Specifying the parameter prevents the first packet from being dropped. All packet sizes are rounded up to a multiple of the system page size. The default packet size *max* value is **`8192`**.

traceoptions *options* ;

Tracing options for EGP (can be overridden on a group or neighbor basis):

`packets`	All EGP packets, or packets *[***`detail`***]* send or *[***`detail`***]* **`recv`** (**`detail`** provides a more verbose format to provide more details; if used, **`detail`** must come before **`send`** or **`recv`**)
`hello`	EGP HELLO/I-HEARD-U packets used to determine neighbor reachability
`acquire`	EGP ACQUIRE/CEASE packets used to initiate and terminate EGP sessions
`update`	EGP POLL/UPDATE packets used to request and receive reachability updates

Group Clause

group
  [peeras ASnumber]
  [localas ASnumber]
  [maxup number
{ **neighbor *host***
    [metricout metric]
    [preference value]
    [preference2 value]
    [ttl ttl]
    [nogendefault]

[*importdefault*]
[*exportdefault*]
[*gateway *gateway]
[*lcladdr *local-address]
[*sourcenet *network]
[*minhello | p1 *time]
[*minpoll | p2 *time]
***[*traceoptions *options]* ; } ;**

EGP neighbors must be members of a group, which groups all neighbors in one AS. Parameters specified in the `group` clause apply to all the subsidiary neighbors, unless explicitly overridden on a `neighbor` clause. Any number of `group` clauses can specify any number of `neighbor` clauses. You can specify any parameters from the `neighbor` subclause on the `group` clause to provide defaults for the whole group (which you can override for individual neighbors).

The **`group`** clause is the only place to set the following attributes:

peeras *ASnumber*

AS number expected from peers in the group. Learned dynamically.

localas *ASnumber*

AS that GateD represents to the group. Usually only used when masquerading as another AS. Use is discouraged. Set globally in **`autonomoussystem`**.

maxup *number*

Number of neighbors GateD should acquire from this group. GateD attempts to acquire the first **`maxup`** neighbors in the order listed. If one of the first neighbors is not available, it acquires one farther down the list. If after startup, GateD does manage to acquire the more desirable neighbor, it drops the less desirable one. By default, GateD acquires all neighbors in the group.

Group Neighbor Clause

egp yes | no | on | off
[{ preference *value* ;
defaultmetric *metric* ;
packetsize *max* ;
traceoptions *options* ;
group
[*peeras *ASnumber]
[*localas *ASnumber]
[*maxup *number
{ neighbor *host*
[*metricout *metric]
[*preference *value]
[*preference2 *value]
[*ttl *ttl]
[*nogendefault*]
[*importdefault*]

[*exportdefault*]
[*gateway *gateway]
[*lcladdr *local-address]
[*sourcenet *network]
[*p1 *time* | minhello*]
[*p2 *time* | minpoll*]
***[*traceoptions *options]* ; } ; }*]* ;**

Each `neighbor` subclause defines one EGP neighbor within a group. The only required part of the subclause is the host argument, the symbolic host name or IP address of the neighbor.

metricout *metric*

Metric used for all routes sent to this neighbor. Overrides the default metric set in the **`egp`** statement and any metrics specified by export policy, but only for this specific neighbor or group of neighbors.

preference *value*

Preference used for routes learned from these neighbors. Can differ from the default EGP preference set in the **`egp`** statement, so that GateD can prefer routes from one neighbor, or group of neighbors, over another. Import policy can explicitly override this.

preference2 *value*

Tie-breaker, in the case of a preference tie. The default *value* is **`0`**.

ttl *ttl*

IPL time-to-live. Provided when attempting to communicate with improperly functioning routers that ignore packets sent with a TTL 1. The default *ttl* for local neighbors is **`1`**; the default for nonlocal neighbors is **`255`**.

nogendefault

Does not generate a default route when EGP receives a valid update from its neighbor. The default route is only generated when you enable the **`gendefault`** option.

importdefault

Accepts the default route (0.0.0.0) if included in a received EGP update. For efficiency, some networks have external routers announce a default route to avoid sending large EGP update packets. The default route in the EGP update is ignored.

exportdefault

Includes the default route (0.0.0.0) in EGP updates sent to this EGP neighbor. Allows the system to advertise the default route using EGP. Normally a default route is not included in EGP updates.

gateway *gateway*

Router on an attached network used as the next hop router for routes received from this neighbor if a network is not shared with a neighbor. Rarely used.

lcladdr *local-address*

Address used on the local end of the connection with the neighbor. The local address must be on an interface shared with the neighbor, or with the neighbor's gateway when using the **`gateway`** option. A session only opens when an interface with the appropriate local address (through which the neighbor or gateway address is directly reachable) is operating.

sourcenet *network*

Network queried in the EGP Poll packets. If there is no network shared with the neighbor, specify one of the networks attached to the neighbor. Also use to specify a network shared with the neighbor, other than the one on which the EGP packets are sent. Normally not needed. The default is the network shared with the neighbor's address.

p1 *time* or **minhello**

Minimum acceptable interval between the transmission of EGP HELLO packets. If the neighbor fails to respond to three hello packets, GateD stops trying to acquire the neighbor. Setting a larger interval gives the neighbor a better chance to respond. The **`minhello`** is an alias for the **`p1`** value defined in the EGP specification. The default *time* value is **`30`**.

```
egp  yes | no | on | off
   [{   preference value ;
        defaultmetric metric ;
        packetsize max ;
        traceoptions options ;
        group
            [peeras ASnumber]
            [localas ASnumber]
            [maxup number
      {  neighbor host
            [metricout metric]
            [preference value]
            [preference2 value]
            [ttl ttl]
            [nogendefault]
            [importdefault]
            [exportdefault]
            [gateway gateway]
            [lcladdr local-address]
            [sourcenet network]
            [minhello | p1 time]
            [minpoll | p2  time]
            [traceoptions options] ; } ; }] ;
```

p2 *time* or **minpoll**

Time interval between polls to the neighbor. If three polls are sent without a response, the neighbor is declared "down" and all routes learned from that neighbor are removed from the routing database. A longer polling interval supports a more stable routing database but is not as responsive to routing changes. The **`minpoll`** is an alias for the **`p2`** value defined in the EGP specification. The default time *value* is **`120`**.

traceoptions *options*

Tracing options for this EGP neighbor, which are:

`packets`	All EGP packets, or packets *[***`detail`***]* **`send`** or *[***`detail`***]* **`recv`** (**`detail`** provides a more verbose format to provide more details; if used, **`detail`** must come before **`send`** or **`recv`**)
`hello`	EGP HELLO/I-HEARD-U packets used to determine neighbor reachability
`acquire`	EGP ACQUIRE/CEASE packets used to initiate and terminate EGP sessions
`update`	EGP POLL/UPDATE packets used to request and receive reachability updates

bgp

The Border Gateway Protocol (BGP) is an exterior routing protocol used to exchange routing information between multiple transit Autonomous Systems (ASs) as well as between transit and stub ASs. BGP is related to EGP but operates with more capability, greater flexibility, and less bandwidth required. BGP uses path attributes to provide more information about each route. It maintains an AS path, which includes the AS number of each AS the route transits, providing information sufficient to prevent routing loops in an arbitrary topology. You can also use path attributes to distinguish between groups of routes to determine administrative preferences. This allows greater flexibility in determining route preference to achieve a variety of administrative ends.

BGP supports two basic types of sessions between neighbors—internal (sometimes called IBGP) and external. Internal sessions run between routers in the same AS, while external sessions run between routers in different ASs. When sending routes to an external peer, the local AS number is prepended to the AS path. Hence routes received from an external peer are guaranteed to have the AS number of that peer at the start of the path. Routes received from an internal neighbor do not generally have the local AS number prepended to the AS path. Hence, these routes generally have the same AS path the route had when the originating internal neighbor received the route from an external peer. Routes with no AS numbers in the path may be legitimately received from internal neighbors; these indicate that the received route should be considered internal to your own AS.

The BGP implementation supports three versions of the BGP protocol—versions 2, 3 and 4. BGP versions 2 and 3 are similar in capability and function. They only propagate classed network routes, and the AS path is a simple array of AS numbers. BGP version 4 propagates fully general address-and-mask routes, and the AS path has some structure to represent the results of aggregating dissimilar routes.

External BGP sessions may or may not include a single metric, which BGP calls the Multi-Exit Discriminator (MED), in the path attributes. For BGP versions 2 and 3 this metric is a 16-bit unsigned integer; for BGP version 4 it is a 32-bit unsigned integer. In either case, smaller values of the metric are preferred. Currently this metric only breaks ties between routes with equal preference from the same neighbor AS. Internal BGP sessions carry at least one metric in the path attributes, which BGP calls the LocalPref. The size of the metric is identical to the MED. For BGP versions 2 and 3, this metric is better when its value is smaller; for version 4 it is better when it is larger. BGP version 4 sessions optionally carry a second metric on internal sessions, this being an internal version of the MED. The use of these metrics depends on the type of internal protocol processing specified.

BGP collapses routes with similar path attributes into a single update for advertisement. Routes received in a single update are readvertised in a single update. The churn caused by the loss of a neighbor is minimized, and the initial advertisement sent during peer establishment is maximally compressed. BGP does not read information from the kernel message by message, but fills the input buffer. It processes all complete messages in the buffer before reading again. BGP also does multiple reads to clear all incoming data queued on the socket. This feature may cause other protocols to be blocked for prolonged intervals by a busy peer connection.

All unreachable messages are collected into a single message and sent prior to reachable routes during a flash update. For these unreachable announcements, the next hop is set to the local address on the connection, no metric is sent, and the path origin is set to incomplete. On external connections the AS path in unreachable announcements is set to the local AS; on internal connections the AS path is set to zero length.

BGP implementation expects external peers to be directly attached to a shared subnet, and expects those peers to advertise next hops that are host addresses on that subnet (although this constraint can be relaxed by configuration for testing). For groups of internal peers, however, there are several alternatives that can be selected by specifying the group type. Type internal groups expect all peers to be directly attached to a shared subnet so that, like external peers, the next hops received in BGP advertisements may be used directly for forwarding. Type routing groups instead determine the immediate next hops for routes, by using the next hop received with a route from a peer as a forwarding address, and using this to look up an immediate next hop in an IGP's routes. Such groups support distant peers, but need to be informed of the IGP whose routes they use to determine immediate next hops. Finally, type IGP groups expect routes from the group peers not to be used for forwarding at all. Instead, they expect that copies of the BGP routes are also received through an IGP, and that the BGP routes are only used to determine the path attributes associated with the IGP routes. Such groups also support distant peers and also need to be informed of the IGP with which they are running.

For internal BGP group types (and for test groups), where possible, a single outgoing message is built for all group peers based on the common policy. A copy of the message is sent to every peer in the group, with possible adjustments to the next hop field as appropriate to each peer. This minimizes the computational load of running large numbers of peers in these types of groups. BGP allows unconfigured peers to connect if an appropriate group was configured with an **`allow`** clause.

Format

bgp yes | no | on | off
  [{ **preference *value* ;**
    defaultmetric *metric* ;
    traceoptions *options* ;
    group type
      external peeras *ASnumber*
      | internal peeras *ASnumber*
      | igp peeras *ASnumber* proto proto
      | routing peeras *ASnumber* proto *proto* interface *list*
      | test peeras *ASnumber*
   { allow
     { *network*
      network* mask *mask
      network* masklen *number
      all
      host *host* } ;
      peer *host*
        [*metricout *metric]
        [*localas *ASnumber]
        [*nogendefault*]
        [*gateway *gateway]

[preference *value*]
[preference2 *value*]
[lcladdr *local-address*]
[holdtime *time*]
[version *number*]
[passive]
[sendbuffer *number*]
[recvbuffer *number*]
[indelay *time*]
[outdelay *time*]
[keep [all | none]]
[analretentive]
[noauthcheck]
[noaggregatorid]
[keepalivesalways]
[v3asloopokay]
[nov4asloop]
[logupdown]
[ttl *ttl*]
[traceoptions *options*] ;
} ;
}] ;

Options and Parameters

yes | on
no | off (default)

Enables or disables BGP support. Disabled by default.

preference *value* ;

Preference for routes learned from BGP. A preference specified on the **`group`** or **`peer`** statements, or by import policy, can override this. The default preference *value* is **`170`**.

defaultmetric *metric* ;

Metric used when advertising routes over BGP. A metric specified on the **`group`** or **`peer`** statements, or in export policy, can override this. The default *metric* is **`65535`**.

traceoptions *options* ;

Tracing options for BGP. May be overridden on a group or peer basis. The trace *options* are:

`packets`	All BGP packets, or packets *[***`detail`***]* **`send`** or *[***`detail`***]* **`recv`** (**`detail`** provides a more verbose format to provide more details; if used, **`detail`** must come before **`send`** or **`recv`**).
open	BGP OPEN packets used to establish a peer relationship

update	BGP UPDATE packets used to pass network reachability information
keepalive	BGP KEEPALIVE packets used to verify peer reachability

Group Type Clause

peeras

For **`group type`**, specify one of the following **`peeras`** options:

`external peeras` *`ASnumber`*	In the classic external BGP group, full policy checking is applied to all incoming and outgoing advertisements. The external neighbors must be directly reachable through one of the machine's local interfaces. No metric included in external advertisements and the next hop is computed with respect to the shared interface.
`internal peeras` *`ASnumber`*	Internal group operating where there is no IP-level IGP; for example, an SMDS network or MILNET. All neighbors in this group must be directly reachable over a single interface. All next-hop information is computed with respect to this interface. Import and export policy may be applied to group advertisements. Routes received from external BGP or EGP neighbors are readvertised with the received metric.
`igp peeras` *`ASnumber`*	Internal group that runs in association with an interior protocol. The IGP group examines routes the IGP exports, and sends an advertisement only if the path attributes could not be entirely represented in the IGP tag mechanism. Only the AS path, path origin, and transitive optional attributes are sent with routes. No metric is sent, and the next hop is set to the local address the connection uses. Received internal BGP routes are not used or readvertised. Instead, the AS path information is attached to the corresponding IGP route and the latter is used for readvertisement. Since internal IGP peers are sent only a subset of the routes the IGP exports, the export policy used is the IGP's. There is no need to implement the "don't route from peers in the same group" constraint, since the advertised routes are routes that IGP already exports.

`routing peeras` *`ASnumber`*	Internal group that uses the routes of an interior protocol to resolve forwarding addresses. A type routing group propagates external routes between routers not directly connected, and computes immediate next hops for these routes by using the BGP next hop that arrived with the route as a forwarding address to be resolved using an internal protocol's routing information. In essence, internal BGP is used to carry AS external routes, while the IGP is expected to only carry AS internal routes, and the latter is used to find immediate next hops for the former. The next hop in BGP routes advertised to the type routing peers are set to local address on BGP connection to those peers, as it is assumed a route to this address is propagated over IGP. • **proto** ***proto***--Interior protocol used to resolve BGP route next hops, and can be the name of any IGP in the configuration. • **interface** ***list***--Optionally provides a list of interfaces whose routes are carried over the IGP for which third party next hops can be used instead.
`test peeras` *`ASnumber`*	Extension to external BGP that implements a fixed policy using test peers. Fixed policy and special case code make test peers relatively inexpensive to maintain. Test peers do not need to be on a directly attached network. If GateD and the peer are on the same (directly attached) subnet, the advertised next hop is computed with respect to that network; otherwise the next hop is the local machine's current next hop. All routing information advertised by and received from a test peer is discarded, and all BGP advertiseable routes are sent back to the test peer. Metrics from EGP- and BGP-derived routes are forwarded in the advertisement; otherwise no metric is included.

Group Type Allow Clause

group type
external peeras ***ASnumber***
| internal peeras ***ASnumber***
| igp peeras ***ASnumber*** **proto proto**
| routing peeras ***ASnumber*** **proto** ***proto*** **interface** ***list***
| test peeras ***ASnumber***
{ allow

{ ***network***
network **mask** ***mask***
network **masklen** ***number***
all
host ***host*** **} ;**

Allows peer connections from any addresses in the specified range of network and mask pairs. Configure all parameters for these peers on the group clause. The internal peer structures are created when an incoming open request is received, and destroyed when the connection is broken. (For details on specifying the network/mask pairs, see *Route Filtering*.)

Group Type Peer Clause

Configures an individual peer. Each peer inherits all parameters specified on a group as defaults. You can override these defaults using parameters explicitly specified in the **`peer`** subclause. Allows the following parameters:

metricout ***metric***

Primary metric on all routes sent to the specified peer(s). Overrides the default metric, a metric specified on the group, and any metric specified by export policy.

localas ***ASnumber***

AS that GateD represents to this group of peers. *ASnumber* is set globally in **`autonomoussystem`**.

nogendefault

Does not generate a default route when EGP receives a valid update from its neighbor. The default route is generated only when enabling the **`gendefault`** option.

gateway ***gateway***

If a network is not shared with a peer, specifies a router on an attached network used as the next hop router for routes received from this neighbor. Not needed in most cases.

preference ***value***

Preference used for routes learned from these peers. Can differ from the default BGP preference set in the **`bgp`** statement, so that GateD can prefer routes from one peer, or group of peers, over others. Import policy can explicitly override this.

preference2 ***value***

In the case of a preference tie, can break the tie.

lcladdr ***local-address***

Address used on the local end of the TCP connection with the peer. For external peers, the local address must be on an interface shared with the peer or with the peer’s gateway when using the **`gateway`** parameter. A session with an external peer only opens when an interface with the appropriate local address (through which the peer or gateway address is directly reachable) is operating. For other types of peers, a peer session is maintained when any interface with the specified local address is operating. In either case, incoming connections are only recognized as matching a configured peer if they are addressed to the configured local address.

holdtime *time*

BGP holdtime value to use when negotiating the connection with this peer, in seconds. According to BGP, if GateD does not receive a keepalive, update, or notification message within the period specified in the Hold Time field of the BGP Open message, the BGP connection is closed. The value must be either 0 (no keepalives are sent) or at least **3**.

version *number*

Version of the BGP protocol to use with this peer. If specified, only the specified version is offered during negotiation. Currently supported versions are **2, 3,** and **4**. By default, the highest supported version is used first, and version negotiation is attempted.

passive

Does not attempt active OPENs to this peer. GateD should wait for the peer to issue an open. By default, all explicitly configured peers are active.

sendbuffer *number* and recvbuffer *number*

Controls the amount of send and receive buffering asked of the kernel. The maximum *number* supported is **65535** bytes, although many kernels have a lower limit. Not needed on normally functioning systems. By default, the maximum supported is configured.

indelay *time* and outdelay *time*

Dampens route fluctuations. The **indelay** is the amount of time a route learned from a BGP peer must be stable before it is accepted into the GateD routing database. The **outdelay** is the amount of time a route must be present in the GateD routing database before it is exported to BGP. Default *time* in both cases is **0**.

group type
.
.
.
peer *host*
[*metricout *metric]
[*localas *ASnumber]
[*nogendefault*]
[*gateway *gateway]
[*preference *value]
[*preference2 *value]
[*lcladdr *local-address]
[*holdtime *time]
[*version *number]
[*passive*]
[*sendbuffer *number]
[*recvbuffer *number]
[*indelay *time]
[*outdelay *time]

[* keep *[*all | none*]]
[*analretentive*]
[*noauthcheck*]
[*noaggregatorid*]
[*keepalivesalways*]
[*v3asloopokay*]
[*nov4asloop*]
[*logupdown*]
[*ttl *ttl]
***[*traceoptions *options]* ; } ; }*]* ;**

keep all

Retains routes learned from a peer even if the routes’ AS paths contain one of our exported AS numbers.

analretentive

Issues warning messages when receiving questionable BGP updates such as duplicate routes and/or deletions of nonexistent routes. Normally these events are silently ignored.

noauthcheck

Communicates with an implementation that uses some form of authentication other than the normal authentication field of all ones.

noaggregatorid

GateD should specify the routerid in the **`aggregator`** attribute as zero (instead of its routerid) in order to prevent different routers in an AS from creating aggregate routes with different AS paths.

keepalivesalways

GateD should always send keepalives, even when an update could have correctly substituted for one. Allows interoperability with routers that do not completely obey the protocol specifications on this point.

v3asloopokay

By default, GateD does not advertise routes whose AS path is looped (that have an AS appearing more than once in the path) to version 3 external peers. Setting this flag removes this constraint. Ignored when set on internal groups or peers.

nov4asloop

Does not advertise routes with looped AS paths to version 4 external peers. Can be useful to avoid advertising such routes to peer which would incorrectly forward the routes on to version 3 neighbors.

logupdown

Logs a message using syslog whenever a BGP peer enters or leaves ESTABLISHED state.

ttl *ttl*

Provided when attempting to communicate with improperly functioning routers that ignore packets sent with a TTL 1. Not all kernels allow the TTL to be specified for TCP connections. The default ttl for local neighbors is 1; the default for nonlocal neighbors is **255**.

traceoptions *options* ;

Tracing options for this BGP neighbor include:

packets	All BGP packets, or packets *[***detail***]* **send** or *[***detail***]* **recv** (**detail** provides a more verbose format to provide more details; if used, **detail** must come before **send** or **recv**)
open	BGP OPEN packets used to establish a peer relationship
update	BGP UPDATE packets used to pass network reachability information
keepalive	BGP KEEPALIVE packets used to verify peer reachability

ospf

Open Shortest Path First (OSPF) routing is a shortest-path-first (SPF) or link-state protocol. OSPF is an interior gateway protocol that distributes routing information between routers in a single Autonomous System (AS). OSPF chooses the least cost path as the best path. Suitable for complex networks with many routers, OSPF provides equal cost multipath routing where packets to a single destination can be sent over more than one interface simultaneously. In a link-state protocol, each router maintains a database describing the entire AS topology, which it builds out of the collected link state advertisements of all routers. Each participating router distributes its local state (that is, the router's usable interfaces and reachable neighbors) throughout the AS by flooding.

Each multiaccess network with at least two attached routers has a designated router and a backup designated router. The designated router floods a link state advertisement for the multiaccess network and has other special responsibilities. The designated router concept reduces the number of adjacencies required on a multiaccess network.

OSPF lets you group networks into areas. Routing information passed between areas is abstracted, which can significantly reduce routing traffic. OSPF uses four different types of routes, listed in order of preference—intra-area, inter-area, type 1 external, and type 2 external. Intra-area paths have destinations within the same area, while inter-area paths have destinations in other OSPF areas. AS External (ASE) routes are routes to destinations external to the AS. Routes imported into OSPF as type 1 routes are supposed to be from IGPs whose external metrics are directly comparable to OSPF metrics.

When making a routing decision, OSPF adds the internal cost of the AS Border router to the external metric. Type 2 ASEs are used for EGPs whose metrics are not comparable to OSPF metrics. In this case, GateD uses only the internal OSPF cost of the AS Border router in the routing decision.

From the topology database, each router constructs a tree of the shortest paths with itself as the root. This shortest-path tree gives the route to each destination in the AS. Externally derived routing information appears on the tree as leaves. The link-state advertisement format distinguishes between information acquired from external sources and from internal routers, so that there is no ambiguity about the source or reliability of routes. Externally derived routing information (for example, routes learned from EGP or BGP) passes transparently through the AS and is separate from OSPF's internally derived data. Each external route can also be tagged by the advertising router, enabling a passing of additional information between routers on the borders of the AS.

OSPF optionally includes type of service (TOS) routing and allows administrators to install multiple routes to a given destination for each type of service (such as for low delay or high throughput.) A router running OSPF uses the destination address and the TOS to choose the best route to the destination.

OSPF intra- and inter-area routes are always imported into the GateD routing database with a preference of 10. It would be a violation of the protocol if an OSPF router did not participate fully in the area's OSPF, so it is not possible to override this. Although it is possible to give other routes lower preference values explicitly, it is ill-advised to do so.

Hardware multicast capabilities are also used where possible to deliver link-status messages.

OSPF areas are connected by the backbone area, the area with identifier 0.0.0.0. All areas must be logically contiguous and the backbone is no exception. To permit maximum flexibility, OSPF allows the configuration of virtual links to enable the backbone area to appear contiguous when they are actually not.

All routers in an area must agree on that area's parameters. A separate copy of the link-state algorithm is run for each area. Because of this, most configuration parameters are defined on a per area basis. All routers belonging to an area must agree on that area's configuration. Misconfiguration leads to adjacencies not forming between neighbors, and routing information might not flow, or even loop.

Authentication. You can authenticate OSPF protocol exchanges. Authentication guarantees that routing information is imported only from trusted routers, to protect the Internet and its users. There are two authentication schemes available. The first uses a simple authentication key of up to eight characters and is standardized. The second is still experimental and uses the **MD5** algorithm and an authentication key of up to 16 characters.

The simple password provides very little protection, because in many cases it is possible to easily capture packets from the network and learn the authentication key. The experimental MD5 algorithm provides much more protection, as it does not include the authentication key in the packet.

The OSPF specification currently specifies that you configure the authentication type per area with the ability to configure separate passwords per interface. This was extended to allow configuration of different authentication types and keys per interface. Also, you can specify both a primary and a secondary authentication type and key on each interface. Outgoing packets use the primary authentication type, but incoming packets may match either the primary or secondary authentication type and key.

You configure OSPF in the TCPWARE:GATED.CONF file using a GateD protocol statement.

Format

ospf yes | no | on | off
  [{ **defaults**
    { **preference** *value* **;**
      cost *cost* **;**
      tag *[***as***]* *tag* **;**
      type 1 | type 2 ;
  } ;
  exportlimit *routes* **;**
  exportinterval *time* **;**
  traceoptions *options***;**
  monitorauthkey *key* **;**
  monitorauth none | *[***simple | md5***]* *authkey* **;**
    backbone | area *area*
      { **authtype 0 | authtype 1 | none | simple ;**
        stub *[***cost** *cost]* **;**
        networks
        { *network* *[***restrict***]* **;**

```
            network mask mask [restrict] ;
            network masklen number [restrict] ;
            host host [restrict] ;
            } ;
        stubhosts
        {   host cost cost ; } ;
        interface list [cost cost]
        {   interface-parameters } ;
        interface list nonbroadcast [cost cost]
        {   pollinterval time ;
            routers
            {   gateway [eligible] ; } ;
            interface-parameters
            } ;
        /* Backbone only: */
        virtuallink neighborid router-id transitarea area
        {   interface-parameters } ;
    } ;
}] ;
```

Options and Parameters

yes | on
no | off

Enables or disables OSPF support.

defaults

Defaults used when importing OSPF ASE routes into the GateD routing table, and exporting routes from the GateD routing table into OSPF ASEs, including:

preference ***value*;**	How OSPF routes compete with routes from other protocols in the GateD routing table. The default preference *value* is **150**.
cost ***cost*** **;**	Used when exporting a non-OSPF route from the GateD routing table into OSPF as an ASE. Export policy can explicitly override this. The default *cost* is **1**.
tag *[***as***]* ***tag*** ;	OSPF ASE routes have a 32-bit tag field that the OSPF protocol does not use, but export policy can use it to filter routes. When OSPF interacts with an EGP, you can use the tag field to propagate AS path information. In this case you would specify the **as** keyword and the tag is limited to 12 bits of information. The default *tag* value is **0**.
type 1 or 2 ;	Export policy can explicitly change and override the default here. The default is **type 1**.

exportlimit *routes* ;

How many ASEs are generated and flooded in each batch. The default export limits *routes* value is **`100`**.

exportinterval *time* ;

How often a batch of ASE link state advertisements are generated and flooded into OSPF. The default export interval *time* value is **`1`** (once per second).

traceoptions *options* ;

In addition to the following OSPF specific trace flags, OSPF supports the state which traces interface and neighbor state machine transitions:

`lsabuild`	Link State Advertisement creation
`spf`	Shortest Path First (SPF) calculations
`lsatransmit`	Link State Advertisement (LSA) transmission
`lsareceive`	LSA reception
`state`	State transitions

Packet tracing options (which you can modify with **`detail, send,`** and **`recv`**):

`hello`	OSPF HELLO packets used to determine neighbor reachability
`dd`	OSPF Database Description packets used in synchronizing OSPF databases
`request`	OSPF Link State Request packets used in synchronizing OSPF databases
`lsu`	OSPF Link State Update packets used in synchronizing OSPF databases
`ack`	OSPF Link State Ack packets used in synchronizing OSPF databases

monitorauthkey key ;
monitorauth none | *[*simple | md5*]* *authkey* ;

You can query the OSPF state using the **`ospf_monitor`** (this should be a hyperlink) utility, which sends nonstandard OSPF packets that generate a text response from OSPF. If you configure an authentication key, the incoming requests must match the specified authentication key. These packets cannot change OSPF state, but the act of querying OSPF can expend system resources. Not authenticated by default.

backbone/area Clause Options and Parameters

ospf yes | no | on | off
***[*{ defaults**
{ preference *value* ;

```
            cost cost ;
            tag [as] tag ;
            type 1 | type 2 ;
        } ;
        exportlimit routes ;
       exportinterval time ;
       traceoptions options;
             monitorauthkey key ;
             monitorauth none | [simple | md5] authkey ;
             backbone | area area
               {   authtype 0 | authtype 1 | none | simple ;
                   stub [cost cost] ;
                   networks
                    {   network [restrict] ;
                        network mask mask [restrict] ;
                        network masklen number [restrict] ;
                        host host [restrict] ;
                        } ;
                    stubhosts
                    {   host cost cost ; } ;
                     interface list [cost cost]
                    {   interface-parameters } ;
                      interface list nonbroadcast [cost cost]
                      {   pollinterval time ;
                            routers
                       {   gateway [eligible] ; } ;
                        interface-parameters
                        } ;
                  /* Backbone only: */
                   virtuallink neighborid router-id transitarea area
                   {   interface-parameters } ;
            } ;
  }] ;
```

backbone or **area** ***area***

Configures each OSPF router into at least one OSPF area. If you configure more than one area, at least one must be the backbone. Configure the backbone using the **`backbone`** keyword only; you cannot specify it as area 0. The backbone interface can be a **`virtuallink`**.

Further parameters include:

`authtype 0` or **`1`** or **`none`** or **`simple`**	OSPF specifies an authentication scheme per area. Each interface in the area must use this same authentication scheme, although it can use a different authentication key. 0 is the same as **`none; 1`** is the same as **`simple`**.

stub *[***cost** ***cost****]*	A stub area is one in which there are no ASE routes. Use cost to inject a default route into the area with the specified cost.
networks **{** ***network [*restrict*]* ;** ***network*** **mask** ***mask [*restrict*]* ;** ***network*** **masklen** ***number [*restrict*]* ;** **host** ***host [*restrict*]* ; } ;**	The **networks** list describes the scope of an area. Intra-area LSAs that fall within the specified ranges are not advertised into other areas as inter-area routes. Instead, the specified ranges are advertised as summary network LSAs. If you specify **restrict**, the summary network LSAs are not advertised. Intra-area LSAs that do not fall into any range are also advertised as summary network LSAs. This option is very useful on well designed networks in reducing the amount of routing information propagated between areas. The entries in this list are either networks, or a subnetwork/mask pair.
stubhosts { ***host*** **cost** ***cost*** **; }**	The stubhosts list specifies directly attached hosts that should be advertised as reachable from this router, and the costs with which they should be advertised. Specify point-to-point interfaces here on which it is not desirable to run OSPF. It is also useful to assign an additional address to the loopback interface (one not on the 127 network) and advertise it as a stub host. If this address is the same one used as the router ID, it enables routing to OSPF routers by router ID, instead of by interface address. This is more reliable than routing to one of the router's interface addresses, which may not always be reachable.
interface ***list*** **cost** ***cost*** **{*interface-parameters*}**	Use this form of the **interface** clause (with the optional *cost* value, and immediately followed by the ***interface-parameters***) to configure a broadcast (which requires IP multicast support) or a point-to-point interface. (See the **interfaces** statement for a description of *list*.) Each interface has a cost. The costs of all the interfaces a packet must cross to reach a destination are summed to get the cost to that destination. The *cost* can be any nonzero value (the default is **1**).

The following are the ***interface-parameters***. You can specify them on any class of interface:

```
enable | disable ;
retransmitinterval time ;
transitdelay time ;
priority value ;
hellointerval time ;
routerdeadinterval time ;
authkey key ;
```

retransmitinterval ***time***	Number of seconds between link state advertisement retransmissions for adjacencies belonging to this interface.
transitdelay ***time***	Estimated number of seconds required to transmit a link state update over this interface. Takes into account transmission and propagation delays and must be greater than 0.
priority ***value***	Number between 0 and 255 specifying the priority for becoming the designated router on this interface. When two routers attached to a network both attempt to become designated router, the one with the highest priority prevails. A router whose router priority is 0 is ineligible to become designated router.
hellointerval ***time***	Length of time, in seconds, between Hello packets that the router sends on the interface.
routerdeadinterval ***time***	Number of seconds not hearing a router's Hello packets before the router's neighbors will declare it down.
authkey ***key***	Used by OSPF authentication to generate and verify the authentication field in the OSPF header. You can configure the authentication key on a per-interface basis. Specify it using one to eight decimal digits separated by periods, a one to eight byte hexadecimal string preceded by `0x`, or a one to eight character string in double quotes.

ospf yes | no | on | off
.
.
.
backbone | area *area*
{ authtype 0 | authtype 1 | none | simple ;
stub *[*cost cost*]* ;
networks
{ *network* [restrict*]* ;
***network* mask *mask* [restrict*]* ;**
***network* masklen *number* [restrict*]* ;**
host *host* [restrict*]* ;

```
                } ;
            stubhosts
            {   host cost cost ; } ;
             interface list [cost cost]
             {   interface-parameters } ;
              interface list nonbroadcast [cost cost]
             {   pollinterval time ;
                 routers
                 {   gateway [eligible] ; } ;
                  interface-parameters
                  } ;
              /* Backbone only: */
              virtuallink neighborid router-id transitarea area
              {   interface-parameters } ;
           } ;
    }] ;
interface list nobroadcast [cost cost]
    {  pollinterval time ;
       routers
      {  gateway [eligible] ; } ;
         interface-parameters } ;
```

This form of the `interface` clause (with the `nobroadcast` option) is for point-to-point interfaces only. By default, OSPF packets to neighbors on point-to-point interfaces are sent using the IP multicast mechanism. GateD detects this condition and falls back to using sending unicast OSPF packets to this point-to-point neighbor.

If you do not want IP multicasting, because the remote neighbor does not support it, specify **`nobroadcast`** to force the use of unicast OSPF packets. You can also use this option to eliminate warnings when GateD detects the bug mentioned previously. (See the previous page for the ***`interface-parameters`***.)

Use this form of the **`interface`** clause to specify a nonbroadcast interface on a nonbroadcast multiaccess (NBMA) media. Since an OSPF broadcast media must support IP multicasting, you must configure a broadcast-capable media, such as Ethernet, that does not support IP multicasting as a nonbroadcast interface. A nonbroadcast interface supports any of the standard interface clauses listed previously, plus the following two that are specific to nonbroadcast interfaces:

`pollinterval` ***`time`***	Before adjacency is established with a neighbor, OSPF packets are sent periodically at the specified poll interval.
`routers` ***`gateway`***	By definition, it is not possible to send broadcast packets to discover OSPF neighbors on a nonbroadcast, so you must configure all neighbors. The list includes one or more neighbors and an indication of their eligibility to become a designated router.

virtuallink neighborid *routerid* transitarea *area*
{ *interface-parameters* } ;

For backbone only: Virtual links are used to establish or increase connectivity of the backbone area. The **`neighborid`** is the router-ID of the other end of the virtual link. The transit area specified must also be configured on this system. You can specify all standard interface parameters defined by the interface clause previously described on a virtual link. (See the previous page for the ***`interface-parameters`***.)

static

The **`static`** statements define the static routes GateD uses. A single **`static`** statement can specify any number of routes. These statements must occur after **`protocol`** statements and before **`control`** statements in GATED.CONF. Specify any number of **`static`** statements, each containing any number of static route definitions. You can override these routes with ones with better preference values.

Format

static
{ host ***host*** **gateway** ***list***
| ***network [*****mask** ***mask*** **| masklen** ***number]*** **gateway** ***list***
| default gateway ***list***
[**interface** ***list]***
[**preference** ***value]***
[**retain*****]***
[**reject*****]***
[**blackhole*****]***
[**noinstall*****]***
;
network [**mask** ***mask*** **| masklen** ***number]***
interface ***interface***
[**preference** ***value]***
[**retain*****]***
[**reject*****]***
[**blackhole*****]***
[**noinstall*****]***
;
};

Options and Parameters

host...gateway ***list*** or **default gateway** ***list***

Most general form of the static statement. 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, they are limited by the number of multipath destinations supported.

The second form of the ***`network`*** **`mask...`** clause farther down in the statement is for primitive support of multiple network addresses on one interface.

interface ***list***

Gateways are valid only when they are on one of these interfaces.

preference ***value***

Preference of this static route. Controls how this route competes with routes from other protocols.

The default *value* is **`60`**.

retain

Prevent specific static routes from being removed. Normally GateD removes all routes except interface routes from the kernel forwarding table during a graceful shutdown. Useful for ensuring that some routing is available when GateD is down.

reject, blackhole

Not supported in TCPware. Install this route as a reject or blackhole route. Instead of forwarding a packet like a normal route, reject routes drop packets and send **`unreachable`** messages to the packet originators. Not all kernel forwarding engines support reject routes. A blackhole route is like a reject route, except that **`unreachable`** messages are not supported.

noinstall

Do not install the route in the kernel forwarding table when active, but make it still exportable to other protocols. Normally the route with the lowest preference is installed there and is the route exported to other protocols.

import

The control statements are:

- **import**
- **export**
- **aggregate**
- **generate**

Format

import *[*restrict | preference *value]*

The **`import`** statements control importing routes from routing protocols, and installing the routes in GateD's routing database. The format of an **`import`** statement varies depending on the source protocol. In all cases, you can specify one of two keywords to control how routes compete with other protocols:

`restrict`	Restrict the routes from 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 that they will not be installed in the forwarding table or exported to other protocols.
`preference` *`value`*	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 can be exported to other protocols. The individual protocols configure the default preferences.

Importing Routes from BGP and EGP

You can control EGP importation by AS. Note that EGP and BGP versions 2 and 3 only support propagating natural networks, so the host and default route filters are meaningless. BGP version 4 supports propagating any destination along with a contiguous network mask.

EGP and BGP both store any routes rejected implicitly by their not being mentioned in a route filter, or explicitly if **`restrict`** appears 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 removes the need to break and reestablish a session on reconfiguring if changing the importation policy.

The syntax of the **`import`** statement for importing routes from BGP or EGP is any of the following:

import proto bgp | egp autonomoussystem *ASnumber* restrict ;
import proto bgp | egp autonomoussystem *ASnumber*
    ***[*preference *value]* {**
    ***route-filter [*restrict | preference *value]* ; } ;**
import proto bgp aspath *ASpathregexp*

origin any | *[*igp*]* *[*egp*]* *[*incomplete*]* restrict ;
import proto bgp aspath *ASpathregexp*
origin any | *[*igp*]* *[*egp*]* *[*incomplete*]*
***[*preference *value]* {**
route*filter [*restrict | preference *value]* ; } ;

The third and fourth variation of the `import` statements is for BGP only and supports controlling propagation by using AS path regular expressions. An AS path is a list of ASs that routing information passes through to get to a router, and an indicator of the origin of the AS path. Use this information to set the preference of one path to a destination network over another. You do this by listing patterns applied to AS paths when importing and exporting routes. Each AS that a route passes through prepends its AS number to the beginning of the AS path.

Aspath Clause

The following **`aspath`** clause in the **`import`** statement indicates that an AS matching the ***`ASpathregexp`*** with the specified origin is matched. The parameters follow:

aspath *ASpathregexp* origin any | *[*igp*]* *[*egp*]* *[*incomplete*]*

Aspath Clause Regular Expression

ASpathregexp

Regular expression, with the alphabet as the set of AS numbers, consisting of one or more AS path expressions, which are terms and operators. An AS path term (***`ASpathterm`***) consists of the following:

`ASnumber`	Any valid AS system number, from 1 through 65534.
`.`	Matches any AS number.
`(`*`ASpathregexp`***`)`**	Parentheses group sub-expressions. An operator such as asterisk **`(*)`** or question mark **`(?)`** works on a single element or on a regular expression enclosed in parentheses.

Aspath Clause Operators

AS path operators consists of the following:

`ASpathterm` **`{m}`**	Exactly m repetitions, where *m* is a positive integer.
`ASpathterm` **`{m,}`**	*m* or more repetitions, where *m* is a positive integer.
`ASpathterm` **`{m,n}`**	At least *m* and at most *n* repetitions, where *m* and *n* are both nonnegative integers and *m* <= *n*.
`ASpathterm` **`*`**	Zero or more repetitions (shorthand for **`{0,}`**).

ASpathterm **+**	One or more repetitions (shorthand for **{1,}**).
ASpathterm **?**	Zero or one repetition (shorthand for **{0,1}**).
ASpathterm | *ASpathterm*	Matches either term.

Remaining Import Statement Options

origin any | *[*igp*]* *[*egp*]* *[*incomplete*]*

Details the completeness of AS path information. An origin of **igp** indicates that the route was learned from an interior routing protocol and is most likely complete. An origin of **egp** indicates that the route was learned from an exterior routing protocol that does not support AS paths (EGP for example), and that the path is most likely not complete. When the path information is definitely not complete, use **incomplete**.

Importing Routes from RIP, HELLO, and Redirects

You can control importing RIP, HELLO, and Redirect routes by any protocol, source interface, or source gateway. If using more than one, they are processed from most general (protocol) to most specific (gateway). RIP and HELLO do not support preferences to choose between routes of the same protocol; they use metrics instead. They also do not save rejected routes since they have short update intervals.

The syntax of the **import** statement for importing routes from RIP, HELLO, or redirects is either of the following:

import proto rip | hello | redirect
  [*interface *list* | gateway *list]
  restrict ;

import proto rip | hello | redirect
  [*interface *list* | gateway *list]
  [*preference *value]
  { *routefilter [*restrict | preference *value]* ; } ;

Importing Routes from OSPF

You can only control importing AS External (ASE) routes. OSPF intra- and inter-area routes are always imported into the GateD routing table with a **preference** of **10**. If using an **ospftag**, the import clause only applies to routes with the specified tag.

You can only restrict importing OSPF ASE routes if functioning as an AS border router. Do this by specifying an **export ospfase** clause. Specifying an empty export clause can restrict importing ASEs, when no ASEs are exported.

Like the other interior protocols, you cannot use **preference** to choose between OSPF ASE routes; OSPF costs accomplish this. Routes rejected by policy go into the table with a negative preference.

The syntax of the **`import`** statement for importing routes from OSPF is either of the following:

import proto ospfase *[*tag *ospftag]* restrict ;

import proto ospfase *[*tag *ospftag]*
[preference *value]*
{ *routefilter [*restrict | preference *value]* ; } ;

export

The control statements are:

- **import**
- **export**
- **aggregate**
- **generate**

Format

export *[*restrict | metric *metric]*

The **`export`** statement controls which routes GateD advertises to other systems. Like **`import`**, the **`export`** syntax varies slightly for each protocol. Both syntaxes are similar and the meanings of many of the parameters are the same. The main difference is that while source information controls importing routes, both destination and source information control exporting routes.

The outer portion of a given **`export`** statement specifies the destination of the routing information you control. The middle portion restricts the sources. The innermost portion is a route filter used to select individual routes.

One thing that applies in all cases is the specification of a metric. All protocols define a default metric for routes exported. In most cases, this can be overridden at several levels of the export statement. The most specific specification of a metric is the one applied to the route exported. The values you can specify for a metric depend on the destination protocol the **`export`** statement references:

`restrict`	Do not export anything. If specified on the destination portion of the **`export`** statement, it means not to export anything to this destination. If specified on the source portion, it means not to export anything from this source. If specified as part of a route filter, it means not to export the routes matching that filter.
`metric` *`metric`*	Metric used when exporting to the specified destination.

Exporting to EGP and BGP

The AS controls exporting to EGP and BGP, the same policy applied to all routers in the AS. EGP metrics range from 0 through 255, with 0 the most attractive. BGP metrics are 16-bit unsigned quantities (that range from 0 through 65535, inclusive with 0 the most attractive). While BGP version 4 actually supports 32-bit unsigned quantities, GateD does not yet support this.

If you do not specify an export policy, only routes to attached interfaces are exported. If you specify any policy, the defaults are overridden; you should explicitly specify everything you want 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.)

The syntax of the **`export`** statement for exporting routes to EGP or BGP is either of the following:

export proto bgp | egp as *ASnumber* restrict ;

export proto bgp | egp as *ASnumber* [metric *metric]*
{ *exportlist* ; } ;

Exporting to RIP and HELLO

Any protocol, interface, or gateway can control exporting to RIP and HELLO. If you specify more than one, they are processed from most general (protocol) to most specific (gateway). It is not possible to set metrics for exporting RIP routes into RIP, or exporting HELLO routes into HELLO. Attempts to do this are silently ignored.

If you do not specify an export policy, RIP and interface routes are exported into RIP and HELLO, and interface routes are exported into HELLO. If you specify any policy, the defaults are overridden; it is necessary to explicitly specify everything that should be exported.

RIP version 1 and HELLO assume 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 is capable of propagating all routes, when not sending version 1 compatible updates.

To announce routes that specify a next hop of the loopback interface (static and internally generated default routes) over RIP or HELLO, specify the metric at some level in the export clause. Just setting a default metric is not sufficient. This is a safeguard to verify that the announcement is intended.

The syntax of the **`export`** statement for exporting routes to RIP or HELLO is either of the following:

export proto rip | hello
***[*interface *list* | gateway *list]* restrict ;**

export proto rip | hello
[*interface *list* | gateway *list]* [metric *metric]
{ *exportlist* ; } ;

Exporting to OSPF

It is not possible to create OSPF intra- or inter-area 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 (see the OSPF protocol configuration for details on the two types). Specify the default type using the **`defaults`** subclause of the **`ospf`** clause. You can override this with the **`export`** statement.

OSPF ASE routes also have the provision to carry a tag. This is an arbitrary 32-bit number you can use on OSPF routers to filter routing information. (See the OSPF protocol configuration for details on OSPF tags.) You can override the default tag specified by the **`ospf defaults`** clause with a tag specified on the **`export`** statement.

The syntax of the **`export`** statement for exporting routes to OSPF is either of the following:

export proto osfpase *[*type 1 | 2*] [*tag *ospf-tag]* restrict ;

export proto osfpase *[*type 1 | 2*] [*tag *ospf-tag]*
** *[*metric metric*]***
** { exportlist ; } ;**

Exporting BGP and EGP Routes

You can specify BGP and EGP routes by source AS. You can export all routes by AS path. The syntax of the **`proto`** statement for exporting BGP or EGP routes is either of the following:

proto bgp | egp autonomoussystem *ASnumber* restrict ;

proto bgp | egp autonomoussystem *ASnumber* *[*metric metric*]*
{ *routefilter [*restrict | metric *metric]* ; } ;

Exporting RIP and HELLO Routes

You can export RIP and HELLO routes by protocol, source interface, or source gateway. The syntax of the **`proto`** statement for exporting RIP or HELLO routes is either of the following:

proto rip | hello
***[*interface *list* | gateway *list]* restrict ;**

proto rip | hello
[*interface *list* | gateway *list] [*metric *metric]
{ *routefilter [*restrict | metric *metric]* ; } ;

Exporting OSPF Routes

You can export both OSPF and OSPF ASE routes into other protocols. The syntax of the **`proto`** statement for exporting OSPF routes is either of the following:

proto ospfase | ospfase restrict ;

proto ospfase | ospfase *[*metric *metric]*
{ *routefilter [*restrict | metric *metric]* ; } ;

Exporting Routes from Nonrouting Protocols with Interface

If you want GateD to export direct or static routes, or routes learned from the kernel, use the protocol statement or interface statement along with the interface of the next hop in the GateD configuration file. The syntax of the **`proto`** statement for exporting routes from nonrouting protocols with an interface is either of the following:

proto direct | static | kernel
***[interface list]* restrict ;**

proto direct | static | kernel
[interface list] [metric metric]
{ *routefilter [*restrict | metric *metric]* ; } ;

The `proto` statement parameters include:

`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`**. You can export these routes 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.

Exporting Routes from Nonrouting Protocols by Protocol

If you want GateD to export default or aggregate routes, use the protocol statement in the GateD configuration file. The syntax of the **`proto`** statement for exporting routes from nonrouting protocols by protocol is either of the following:

proto default | aggregate restrict ;
proto default | aggregate
[metric metric]
{ route*filter [*restrict | metric *metric]* ; } ;

The `proto` statement parameters include:

`default`	Routes created by the **`gendefault`** option. Use route generation instead.
`aggregate`	Routes synthesized from other routes when using the **`aggregate`** and **`generate`** statements.

Exporting by AS Path

When configuring BGP, all routes get an AS path when added to the routing table. For all interior routes, this AS path specifies IGP as the origin and no ASEs 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. (The AS path regular expression syntax appears in the *Importing Routes from BGP and EGP* subsection.)

The syntax of the **`proto`** statement for exporting by AS path is either of the following:

proto ***proto*** **| all aspath** ***ASpathregexp***
origin any | ***[*****igp*****]*** ***[*****egp*****]*** ***[*****incomplete*****]*** **restrict ;**
proto ***proto*** **| all aspath** ***ASpathregexp***
origin any | ***[*****igp*****]*** ***[*****egp*****]*** ***[*****incomplete*****]*** ***[*****metric** ***metric]***
{ ***routefilter [*****restrict | metric** ***metric]*** **; } ;**

Exporting by Route Tag

Both OSPF and RIP version 2 currently support tags. All other protocols always have a tag of zero. You can select the source of exported routes based on this tag. This is useful when classifying routes by tag when exporting them into a given routing protocol. The syntax of the **`proto`** statement for exporting by route tag is either of the following:

proto ***proto*** **| all all tag** ***tag*** **restrict ;**
proto ***proto*** **| all all tag** ***tag***
[**metric** ***metric]***
{ ***routefilter [*****restrict | metric** ***metric]*** **; } ;**

aggregate

The control statements are:

- **import**
- **export**
- **aggregate**
- **generate**

Use route aggregation to generate a more general route from a specific one. Use it, for example, at an AS border to generate a route to a network to be advertised through EGP, given the presence of one or more subnets of that network learned through RIP. Regional and national networks also use route aggregation to reduce routing information. By carefully allocating network addresses to clients, regional networks can just announce one route to regional networks instead of hundreds. No aggregation occurs unless explicitly requested in an **`aggregate`** statement.

Aggregate routes are not actually used for packet forwarding by the originator of the aggregate route, 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 prevents packets for unknown component routes from following a default route into another network where they would be continuously forwarded back to the border router, until their TTL expires. Sending an unreachable message for a missing piece of an aggregate is only possible on systems that support reject routes, which TCPware does not.

Format

aggregate default | ***network [*****mask** ***mask*** **| masklen** ***number]***
[preference ***value] [*****brief*****]***
{ proto ***[*****all | direct | static | kernel | aggregate | proto*****]***
[**as** ***AS*** **| tag** ***tag*** **| aspath** ***ASpathregexp]*** **restrict ;**
proto ***[*****all | direct | static | kernel | aggregate | proto*****]***
[**as** ***AS*** **| tag** ***tag*** **| aspath** ***ASpathregexp]*** **[preference** ***value]***
{ ***routefilter [*****restrict | preference** ***value]*** **; } ;**
} ;

Options and Parameters

preference ***value***

The default preference *value* is **`130`**.

brief

Truncate the AS path 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 ***proto***

In addition to the special protocols listed, you can select the contributing protocol from among those currently configured in GateD.

aggregate default | ***network [*****mask** ***mask*** **| masklen** ***number***

[**preference** ***value] [*****brief*****]***
{ proto ***[*****all | direct | static | kernel | aggregate | proto*****]***
[**as** ***AS*** **| tag** ***tag*** **| aspath** ***ASpathregexp]***
restrict ;
proto ***[*****all | direct | static | kernel | aggregate | proto*****]***
[**as** ***AS*** **| tag** ***tag*** **|** ***aspath ASpathregexp]***
[**preference** ***value]***
{ ***routefilter [*****restrict | preference** ***value]*** **; } ;**
} ;

as ***AS***

Restrict selection of routes to those learned from the specified AS.

tag ***tag***

Restrict selection of routes to those with the specified tag.

aspath ***ASpathregexp***

Restrict selection of routes to those that match the specified AS path.

restrict

Restrict certain routes from contributing to the specified aggregate.

A route can only contribute to an aggregate route that is more general than itself; it must match the aggregate under its mask. Any given route can only contribute to one aggregate route, which will be the most specific configured, but an aggregate route can contribute to a more general aggregate.

generate

The control statements are:

- **import**
- **export**
- **aggregate**
- **generate**

A slight variation on aggregation is generating a route based on 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 is to generate a default based on the presence of a route from a peer on a neighboring backbone.

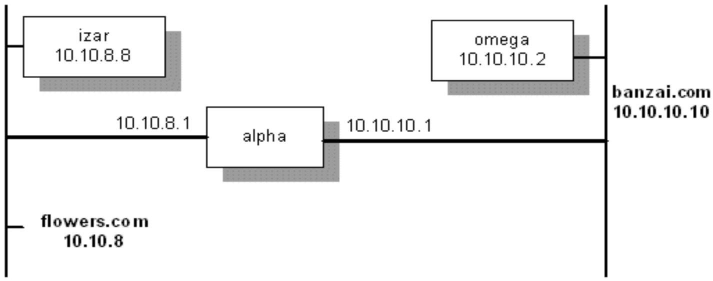
Format

generate default | ***network [*****mask** ***mask*** **| masklen** ***number]***
[preference ***value] [*****brief*****]***
{ *[*as ***AS*** **| tag** ***tag*** **| aspath** ***ASpathregexp]***
restrict ;
proto *[*all | direct | static | kernel | aggregate | proto]
[**as** ***AS*** **| tag** ***tag*** **| aspath** ***ASpathregexp]***
[**preference** ***value]***
{ *routefilter [*restrict | preference ***value]*** **; } ;**
} ;

Sample GateD Configurations

Figure 8-3 shows two networks connected within an AS using RIP. Example 8-1, Example 8-2, and Example 8-3 show the RIP statements on each end host and gateway `alpha,` which has IP forwarding enabled. All systems are running GateD.

Figure 8-3   Sample RIP Configuration



Example 8-1   GateD Configuration File for `izar`

```
#  turn on RIP and listen for updates.
#
rip on;
```

Example 8-2   GateD Configuration File for alpha

```
#  turn on RIP.
#
rip yes;
#
#  use RIP to pass routing information to the banzai network.
#
export proto rip interface 10.10.10.1
    {
    #  we know about the flowers network, so announce it.
    #
    proto direct {
      10.10.8.0 mask 255.255.255.0;
      };
    #  use RIP to announce all routes learned from flowers.
    #
    proto rip interface 10.10.8.0 {
       all;
       };
     };
```

Example 8-3 GateD Configuration File for omega

```
#  turn on RIP and listen for updates.
#
rip on;
```

Example 8-4 shows a sample RIP statement where the gateway announces a default route to the backbone, and announces all of the individual subnet routes to the outside world.

Example 8-4 Default RIP Announcements

```
#  enable RIP:
#
rip yes;

#  using RIP, announce all local subnets via interface 192.168.12.3:
#
export proto rip interface 192.168.12.3 metric 3
    {
       proto rip interface 192.168.1.5
       {
           all;
       };
     };
#
#  Using RIP, announce default via interface 192.168.1.5:
#
export proto rip interface 192.168.3.1
     {
        proto rip interface 192.168.1.5
         {
            default;
         };
      };
```

Example 8-5 shows a configuration for AS 283 that enables RIP and OSPF, which you can use to test both.

Example 8-5 Using RIP and OSPF

```
# this interface is passive:
#
interfaces {
        interface SVA-0 passive;
        };
#
#  this Autonomous System number is 283:
```

```
#
autonomoussystem 283;
#
#  turn on RIP:
#    packets are to be broadcast.
#    metric for routes learned via other protocols is 5.
#    multicast RIP V2 packets on SVA-0.
#
rip yes {
     broadcast;

                       defaultmetric 5;
                       interface SVA-0 version 2 multicast;
                       };
#
# turn on OSPF:
#    Trace Link State Advertisement creation and
#    Shortest Path First calculations
#    use authentication key "ZZZZZZZZ" when handling OSPF queries.
#    this system is on the backbone.
#      use simple password authentication for this area.
#      make this system very unlikely to be a designated router.
#      set the OSPF header authentication key to "YYYYYYYY" for
#      packets going out on SVA-0.
#

ospf yes {
       traceoptions lsabuild spf;
       monauthkey "ZZZZZZZZ";
       backbone {
           authtype simple;
           interface all {
              priority 2;
              };
           interface SVA-0 {
              authkey "YYYYYYYY";
              };
           };
      };
```

PART IV Managing Time Services

Chapter 9

Network Time Protocol (NTP)

Introduction

This chapter describes how to configure and manage the Network Time Protocol (NTP) to synchronize timekeeping among a set of distributed time servers and clients. The synchronization is totally transparent to users.

TCPware's NTP also includes two standard query programs, NTPQ and XNTPDC, the utility NTPTRACE, and the NTPDATE program.

TCPware's NTP implementation is based on Network Time Protocol Version 4 from David L. Mills and the University of Delaware, and complies with RFC 1305. (Copyright information is included on the verso of the title page of this guide.)

This chapter is divided into the following sections:

- NTP functions
- Implementing NTP
- Timekeeping hosts
- Determining peer hosts
- Modifying the NTP configuration file
- Basic configuration commands
- Advanced configuration
- Basic configuration example
- Troubleshooting NTP
- NTPQ
- XNTPDC
- NTPDATE
- NTPTRACE

NTP Functions

The Network Time Protocol (NTP) is used to synchronize the time of a computer client or server to another server or reference time source, such as a radio, satellite receiver, or modem. It is accurate typically within a millisecond on LANs and up to a few tens of milliseconds on WANs, relative to Coordinated Universal Time (UTC), as provided by a Global Positioning Service (GPS) receiver. Typical NTP configurations use multiple redundant servers and diverse network paths to achieve high accuracy and reliability. TCPware's NTP implementation also supports cryptographic authentication to prevent accidental or malicious protocol attacks.

Synchronized timekeeping involves resolving the frequency difference (or *skew*) and time difference (or *offset*) between clocks in the network. The goal is for hosts to have accurate system time stamps and send accurate time quotes to each other. NTP produces the following data and uses it in its synchronization effort:

Clock offset	Amount to adjust the local clock with the reference clock
Round trip delay	Ability of the local host to send a message to the reference clock so that it arrives at a certain time
Dispersion	Maximum error of the local clock relative to the reference clock

Timekeeping systems are judged based on the following criteria:

Stability	How well a clock can maintain a constant frequency
Accuracy	How well a clock's time compares with national standards
Precision	How precisely a timekeeping system can maintain stability and accuracy
Reliability	How long a timekeeping system is connected and operational

NTP makes local system time adjustments by either *slewing* or *stepping* the clock. Slewing runs the clock faster or slower than its normal frequency to keep the clock at the correct time. Stepping sets the clock immediately to the correct time. Stepping occurs very infrequently, only when there is a large time error to adjust, such as when starting NTPD or when making daylight savings time (DST) changes.

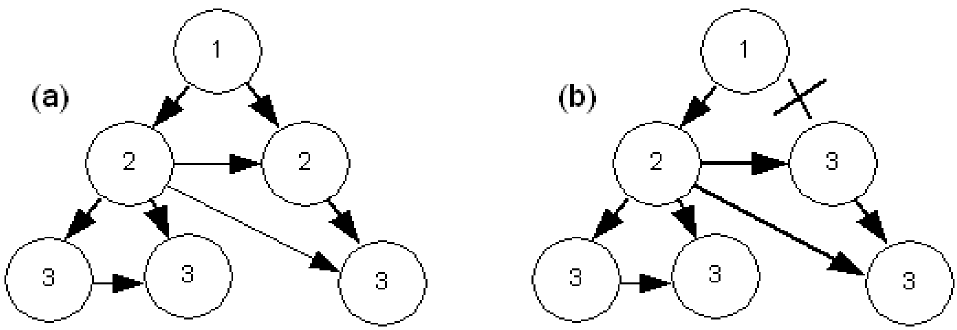
NTP data is exchanged periodically between hosts as encapsulated in UDP datagrams, and adjustments are made based on an NTP algorithm. The frequency of exchange is related to the effort required to synchronize the clocks. Whereas resolving a clock offset may take no more than a few exchanges, resolving skew and maintaining local time to within a millisecond can take hours and dozens of measurements. However, the frequency of exchange is rarely intrusive to normal network operation. Also, the unreliability of UDP has no measurable impact on the process, and the process does not depend on any such reliability.

Synchronized hosts, known as *peers*, are either time servers or clients. Peers are identified by relative NTP *strata* numbers. Lower strata peers act as time servers while higher strata peers are clients who adjust their time clocks according to these servers. An Internet Time Server (ITS) on the network is assigned *stratum 1* because it has radio-clock-generated time based on Universal Coordinated Time (UTC). NTP peers can be (and often are) other types of systems running NTP, not just OpenVMS systems.

Each host has its identifying stratum number encoded within the UDP datagram. This means that hosts can effectively negotiate server and client roles.

The stratum method allows for backup timekeeping in case a node or connection goes down, and stratum numbers may change as a consequence. In Figure 9-1 (a), each node has a stratum number based on hop count, with the ITS at the top of the pyramid. The solid arrows are the active synchronization paths and direction of timing information flow; the lighter arrows are background synchronization paths where timing information is exchanged but not necessarily used for synchronization. Figure 9-1 (b) shows the same network with one of the connections broken—note that the stratum for the affected peer increases from 2 to 3.

Figure 9-1 Synchronization Through Strata



NTP Files

NTP uses the following files:

NTP.CONF?	You maintain an NTP configuration file, TCPWARE:NTP.CONF, of participating peers. You determine the peer hosts with which the local host should negotiate and synchronize, then populate the file with a list of these peers, some of which may be at a higher or lower stratum than the local host. To add stability, multiple NTP servers (systems at a lower stratum) should show up as peers. You need to restart NTP after you edit the NTP.CONF file. (See *Modifying the NTP Configuration File*.)
NTP.KEYS	You maintain a TCPWARE:NTP.KEYS file if access authorization is desired. (See *Authentication Using a Keys File*.)

NTPSERVER.LOG	The NTP server outputs log information to the TCPWARE:NTPSERVER.LOG file. (See *Troubleshooting*.)
NTP.DRIFT	The TCPWARE:NTP.DRIFT file consists of a single line containing a floating point number that records the frequency offset, measured in parts-per-million (PPM). NTP updates this file once an hour to maintain an accurate frequency offset when restarting the NTP service. You should not need to modify this file.

Implementing NTP

To implement NTP:

1	Determine the hosts (peers) with which the local system should negotiate for synchronization. (See *Timekeeping Hosts and Determining Peer Hosts*.)
2	Configure the NTP configuration file by adding these peers to it. (See *Modifying the NTP Configuration File*.)
3	Use one of the query programs.

Timekeeping Hosts

Internet Time Servers (ITSs) connected to accurate radio clocks maintain timekeeping on the Internet. These ITSs are at the absolute lowest level, or stratum 1, of the NTP network. Most timekeeping hosts on a specific network are at stratum 2 or higher.

The NTP hosts negotiate for accurate time by exchanging NTP packets to determine:

1	The stratum each host is on (and, therefore, which one is the time server).
2	The time offset, so that the local host can adjust its time accordingly.

All NTP times are an offset of Universal Coordinated Time (UTC), formerly Greenwich Mean Time (GMT).

Determining Peer Hosts

Determine which list of peers you want to include in the configuration file. These are hosts with which you regularly exchange data and where accurate time coordination is an important factor. Include at least one (but preferably two) peer hosts that you are assured:

- Provide accurate time

- Synchronize to ITSs (if they are not themselves ITSs)

Two hosts provide reliability in case one goes down. You do not need to identify what stratum each peer is on. NTP determines this through the reference information it sends in its UDP packet exchanges.

A list of public NTP servers, along with guidelines for their use, has typically been available at the following Web site:

`http://www.eecis.udel.edu/~mills/ntp/servers.html.`

Modifying the NTP Configuration File

To configure only NTP, you can enter **`@TCPWARE:CNFNET NTP`**. Then add entries to the TCPWARE:NTP.CONF peer configuration file. The commands you add to the file have the syntax and meaning described in the following *Basic Configuration Commands* and *Advanced Configuration* sections.

Basic Configuration Commands

peer *address* [key *key*] [version *version*] [prefer]
server *address* [key *key*] [version *version*] [prefer] [mode *mode*]
broadcast *address* [key *key*] [version *version*] [ttl *ttl*]

These three commands specify the time server *address* to be used and the mode in which to operate. The *address* can be a domain name or an IP address in dotted quad notation.

Command	**Specifies that the local server is to operate in...**
`peer`	Symmetric active mode with the remote server. In this mode, the local server can be synchronized to the remote server and, in addition, the remote server can be synchronized by the local server. This is useful in a network of servers where, depending on various failure scenarios, either the local or remote server may be the better source of time.
`server`	Client mode with the specified remote server. In this mode, the local server can be synchronized to the remote server, but the remote server can never be synchronized to the local server.
`broadcast`	Broadcast mode, where the local server sends periodic broadcast messages to a client population at the broadcast/multicast address specified. Ordinarily, this specification applies only to the local server operating as a sender; for operation as a broadcast client, see the **`broadcastclient`** command that follows. In this mode, address is usually the broadcast address on (one of) the local networks.

The **peer, server**, and **broadcast** options include:

Option	Description
key *key*	All packets sent to an address are to include authentication fields encrypted using the specified key identifier, which is an unsigned 32-bit integer. The default is to not include an encryption field.
version *version*	Specifies the version number to be used for outgoing NTP packets. Versions 1, 2, and 3 are the choices, with version 3 the default.
prefer	Marks the server as preferred. The host is chosen for synchronization among a set of correctly operating hosts.
ttl *ttl*	This option is used only with broadcast mode. It specifies the time-to-live (TTL) to use on multicast packets. The default TTL is 127.

master-clock *stratum*

The **master-clock** command should be used to configure a host that has special hardware to synchronize its clock, or that has its time synchronized by an outside source such as the Compaq Time Synchronization Service (CTSS). NTP propagates time information normally, but makes no changes to system time. A master clock would normally be configured at stratum 0 to 8. However, unless the local clock is reliably disciplined by an outside source, a low stratum may disrupt access by other clients to reliable NTP servers.

local-master *stratum*

The **local-master** command should be used for sites not connected to the Internet, or for the purpose of having a backup time source when all normal synchronization sources are unavailable. This allows a machine to use its local clock as a time source while NTP makes changes to system time to compensate for the clock's frequency error. Specify a stratum number between 8 and 15.

broadcastclient

This command directs the local server to listen for broadcast messages at the broadcast address of the local network. The address is the subnet address with the host field bits set to ones. 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, in order to avoid accidental or malicious disruption in this mode, both the local and remote servers should operate using authentication and the same trusted key and key identifier.

slewalways

NTPD normally steps the clock when there is a relatively large time error to adjust. The **slewalways** command directs the local NTP server to always slew the clock. Use this command to avoid an abrupt one hour clock change when daylight savings time (DST) occurs. For DST changes when slewalways is specified, NTPD slews the clock over a period of about 10 hours.

{ enable | disable } auth

Enables the server to synchronize with unconfigured peers only if the peer was correctly authenticated using a trusted key and key identifier. (See *Authentication Using a Keys File*.) The default for this flag is **`disable`**.

{ enable | disable } monitor

Enables the monitoring facility. See the **`monlist`** command of the XNTPDC program for further information. The default for this flag is **`enable`**.

{ enable | disable } stats

Enables the statistics facility. For further information, see *Monitoring Commands*. The default for this flag is **`enable`**.

{ enable | disable } opcom

Enables OPCOM messaging for NTP. The default for this flag is **`enable`**.

Advanced Configuration

Advanced configuration involves configuring a keys file for authentication, viewing statistics, and using access control and various miscellaneous commands.

Authentication Using a Keys File

The NTP standard specifies an extension which provides cryptographic authentication of received NTP packets. This is implemented in NTPD using the MD5 algorithm to compute a digital signature, or message digest. The specification allows any one of possibly four billion keys, numbered with 32-bit key identifiers, to be used to authenticate an association. 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 the file TCPWARE:NTP.KEYS, which should be exchanged and stored using secure procedures. There are three classes of keys involved in the current implementation. One class is used for ordinary NTP associations, another for the NTPQ utility program, and the third for the XNTPDC utility program.

trustedkey ***key [...]***

Specifies the encryption key identifiers which are trusted for the purposes of authenticating peers suitable for synchronization. The authentication procedures require that both the local and remote servers share the same key and key identifier for this purpose, although different keys can be used with different servers. The key arguments are 32-bit unsigned integers. Note that NTP key 0 is fixed and globally known. If meaningful authentication is to be performed, the 0 key should not be trusted.

requestkey *key*

Specifies the key identifier to use with the XNTPDC program, which uses a proprietary protocol specific to this distribution of NTPD. The key argument to this command is a 32-bit unsigned integer. If no **`requestkey`** command is included in the configuration file, or if the keys do not match, such requests are ignored.

controlkey *key*

Specifies the key identifier to use with the NTPQ program, which uses the standard protocol defined in RFC 1305. The key argument to this command is a 32-bit unsigned integer. If no requestkey command is included in the configuration file, or if the keys do not match, such requests are ignored.

Key File Format

For MD5, keys are 64 bits (8 bytes), read from the TCPWARE:NTP.KEYS file. While key number 0 is fixed by the NTP standard (as 64 zero bits) and may not be changed, one or more of the keys numbered 1 through 15 may be arbitrarily set in the keys file.

The keys file uses the same comment conventions as the configuration file. Key entries use a fixed format of the form:

keyno type key

- ***`keyno`*** is a positive integer
- ***`type`*** is a single character **`M`** for the MD5 key format
- ***`key`*** is the key itself

The key is a one-to eight-character ASCII string using the MD5 authentication scheme. Note that both the keys and the authentication schemes (MD5 or DES) must be identical between a set of peers sharing the same key number.

Note that the keys used by the NTPQ and XNTPDC programs are checked against passwords requested by the programs and entered by hand.

Monitoring Commands

NTP includes a comprehensive monitoring facility suitable for continuous, long term recording of server and client timekeeping performance. (See the **`statistics`** command that follows for a listing and example of each type of statistic currently supported.) The monitoring commands are as follows:

statistics { loopstats | peerstats | clockstats } *[...]*

Enables writing of statistics records. Currently, three kinds of statistics are supported:

`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). 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 a 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 hex in the format described in Appendix B.2.2 of the NTP specification RFC 1305. The final three fields show the offset, delay, and dispersion, all in seconds.
rawstats	Enables recording of raw peer statistics information. Each update of the local clock outputs a line of the following form to the file generation set named rawstats: 50528 68596.786 192.42.95.1 198.115.142.54 3067959784.942463 3067961504.783990 3067961504.783990 3067959784.946858 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 address in dotted quad notation. The final four fields show the time stamps for a peer NTP communication.

statsdir *directory-path*

Indicates the full path of a directory where statistics files should be created. This keyword allows the (otherwise constant) **`filegen`** filename prefix to be modified for file generation sets, which is useful for handling statistics logs.

filegen { loopstats | peerstats | clockstats }
[* file *filename] [* type *typename] [* enable | disable *]

Configures setting of the generation fileset name. Generation filesets provide a means for handling files that are continuously growing during the lifetime of a server. Server statistics are an example for such files. Generation filesets provide access to a set of files used to store the actual data.

At most one element of the set is being written to at any one time. The type given specifies when and how data is directed to a new element of the set. This way, information stored in elements of a fileset that are currently unused are available for administrative operations without the risk of disturbing the operation of NTPD. (Also, they can be removed to free space for new data produced.)

For the **`loopstats, peerstats,`** and **`clockstats`** parameters, see the **`statistics`** command.

The following additional parameters apply:

`filename`	This string is directly concatenated to the directory TCPWARE: or the directory prefix specified using the statsdir option explained above. The suffix for this filename is generated according to the type of a fileset.
`typename`	A file generation set is characterized by **`type`** ***`typename`***: **none**--One element of the fileset is used for each, NTPD server. **day**--One file generation set element is created per day. A day is defined as the period between 00:00 and 24:00 UTC. The fileset member suffix consists of a . (dot) and a day specification in the form YYYYMMDD. YYYY is a 4-digit year number (such as 1997). MM is a two digit month number. DD is a two digit day number. Thus, all information written at 10 December 1997 would end up in a file named *`prefix filename`*`.19971210`. **week**--Any fileset member contains data related to a certain week of a year. The term week is defined by computing day-of-year modulo 7. Elements of such a file generation set are distinguished by appending the following suffix to the fileset filename base: a dot, a 4-digit year number, the letter `W`, and a 2-digit week number. For example, information from January 10th, 1997 would end up in a file with suffix `.1992W1`. **month**--One generation fileset element is generated per month. The filename suffix consists of a dot, a 4-digit year number, and a 2-digit month. **year**--One generation file element is generated per year. The filename suffix consists of a dot and a 4-digit year number. **age**--This type of file generation sets changes to a new element of the fileset every 24 hours of server operation. The filename suffix consists of a dot, the letter `a`, and an 8-digit number. This number is taken to be the number of seconds the server is running at the start of the corresponding 24-hour period.
`enable` and **`disable`**	Information is only written to a file generation by specifying **`enable`**; output is prevented by specifying **`disable`**.

Access Control Commands

NTP implements a general purpose address- and mask-based restriction list. The list is sorted by address and by mask, and the list is searched in this order for matches, with the last match found defining the restriction flags associated with the incoming packets. The source address of incoming packets is used for the match, with the 32-bit address combined with the mask associated with the restriction entry and then compared with the entry's address (which was also combined with the mask) to look for a match.

The restriction facility was implemented to conform with the access policies for the original NSFnet backbone time servers. While this facility may be otherwise useful for keeping unwanted or broken remote time servers from affecting your own, it should not be considered an alternative to the standard NTP authentication facility. Source address based restrictions are easily circumvented by a determined hacker.

restrict *numeric-address [* mask *numeric-mask] [* flag *] [...]*

The *numeric-address* argument, expressed in dotted quad form, is the address of a host or network. The *mask* argument, also expressed in dotted quad form, defaults to 255.255.255.255, meaning that the *numeric-address* is treated as the address of an individual host. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and, given the sort algorithm, is always the first entry in the list. Note that, while *numeric-address* is normally given in dotted-quad format, the text string default, with no mask option, can be used to indicate the default entry.

In the current implementation, **`flag`** always restricts access, such that an entry with no flags indicates that free access to the server is to be given. The flags are not orthogonal, in that more restrictive flags often make less restrictive ones redundant. The flags can generally be classed into two categories: those that restrict time service, and those that restrict informational queries and attempts to do run-time reconfiguration of the server.

You can specify one or more of the following flags:

`ignore`	Ignores all packets from hosts which match this entry. If this flag is specified, neither queries nor time server polls are responded to.
`noquery`	Ignores all NTP mode 6 and 7 packets (information queries and configuration requests) from the source. Time service is not affected.
`nomodify`	Ignores all NTP mode 6 and 7 packets that attempt to modify the state of the server (run-time reconfiguration). Queries which return information are permitted.
`noserve`	Ignores NTP packets whose mode is other than 6 or 7. In effect, time service is denied, though queries may still be permitted.
`nopeer`	Provides stateless time service to polling hosts, but does not allocate peer memory resources to these hosts, even if they otherwise might be considered useful as future synchronization partners.
`notrust`	Treats these hosts normally in other respects, but never uses them as synchronization sources.

`limited`	These hosts are subject to limitation of number of clients from the same net. Net in this context refers to the IP notion of net (class A, class B, class C, and so on). Only the first `client_limit` hosts (as set by the **`clientlimit`** command) that showed up at the server, and that were active during the last `client_limit_period` seconds (as set by the **`clientperiod`** command), are accepted. Requests from other clients from the same net are rejected. Only time request packets are taken into account. Query packets sent by the NTPQ and XNTPDC programs are not subject to these limits. A history of clients is kept using the monitoring capability of NTPD. Thus, monitoring is always active as long as there is a restriction entry with the limited flag.
`ntpport`	This is actually a match algorithm modifier, rather than a restriction flag. Its presence causes the restriction entry to be matched only if the source port in the packet is the standard NTP UDP port (123).

Default restriction list entries, with the flags **`ignore, ntpport,`** for each of the local host's interface addresses are inserted into the table at startup, to prevent the server from attempting to synchronize to its own time. However, a default entry is also always present if it is otherwise unconfigured. No flags are associated with the default entry (everything besides your own NTP server is unrestricted).

clientlimit *limit*

Sets the `client_limit` variable that limits the number of simultaneous access-controlled clients. The default value is 3.

clientperiod *period*

Sets the `client_limit_period` variable that specifies the number of seconds after which a client is considered inactive and thus no longer is counted for client limit restriction. The default value is 3600 seconds.

Miscellaneous Command

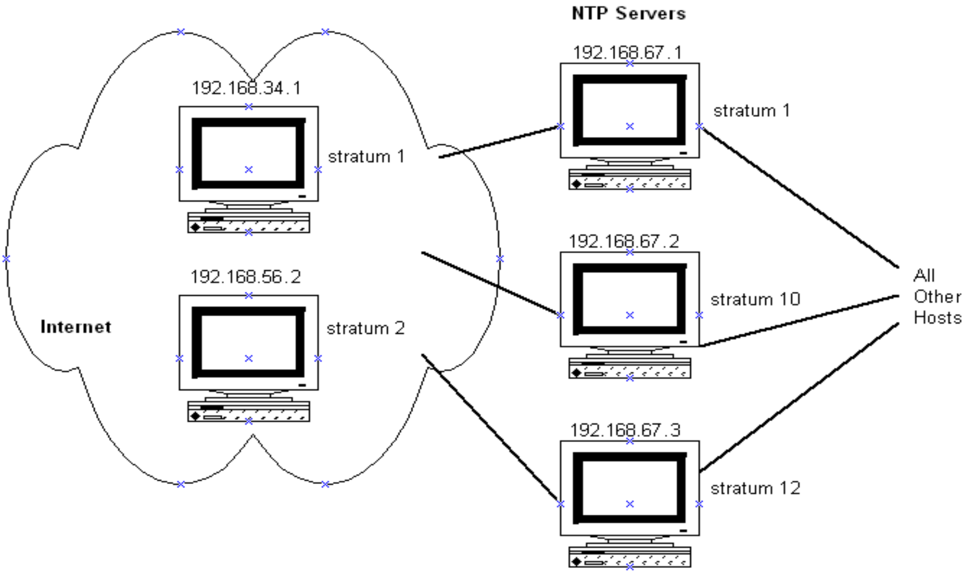
setvar *name[=value] [* default *]*

This command adds an additional system variable. These variables can be used to distribute additional information such as the access policy. If the variable of the form *name = value* is followed by the **default** keyword, the variable is listed as part of the default system variables (**`ntpq rv`** command). These additional variables serve informational purposes only. They are not related to the protocol other than that they can be listed. The known protocol variables always override any variables defined using the **setvar** mechanism.

Basic Configuration Example

Figure 9-2 shows a highly redundant and robust configuration with multiple levels of backups. On the Internet close to your network, you have host 192.168.34.1 running at stratum 1, and 192.168.34.2 at stratum 2. In-house, you have host 192.168.67.1 synchronized with a radio clock and configured as a stratum 1 master clock.

Figure 9-2 Sample NTP Configuration



As backup servers, you have two hosts, 192.168.67.2 and 192.168.67.3, in the climate-controlled room, one configured at stratum 10 and the other at 12. All other workstations on the floor point to these three servers as their synchronization source. When everything is running, every local host is synchronized to 192.168.67.1, since it is closer than Internet host 192.168.34.1. All the machines (peers) run at stratum 2.

If internal host 192.168.67.1 goes down and the Internet connection is still up, either Internet host 192.168.34.1 or 192.168.34.2 is selected depending on its availability, and the backup servers, 192.168.67.2 and 192.168.67.3, run at stratum 2 or 3, depending on which Internet host was selected. The peers synchronize off 192.168.67.2 or 192.168.67.3 at stratum 3 or 4, again depending on which Internet host was selected.

With 192.168.67.1 still unavailable and the Internet connection lost or all the Internet servers unavailable, 192.168.67.2 runs at stratum 10, since it was configured that way as a local clock. It then becomes the lowest stratum number in the network and all other hosts (including 192.168.67.3) are synchronized to it at stratum 11.

If 192.168.67.2 goes down, 192.168.67.3 runs at stratum 12 and all other hosts synchronize at stratum 13. It is important to set the stratum of 192.168.67.3 to 12. If set to 11, it might have a problem synchronizing to 192.168.67.2, since it may try to synchronize off it but finds it has the same stratum value. 192.168.67.3 would rather synchronize to 192.168.67.2 than to itself.

Example 9-1 shows the configuration file entries for each of the three local servers (the other local hosts would all be configured as peers). You do not need to explicitly identify the peer strata, and the order of items is irrelevant.

Example 9-1   Sample Entries in the Host NTP.CONF Files

```
; NTP Configuration on 192.168.67.1
master-clock 1

; NTP Configuration on 192.168.67.2
local-master 10
server 192.168.67.1
server 192.168.34.1
server 192.168.34.2
peer 192.168.67.2


; NTP Configuration on 192.168.67.3
local-master 12
server 192.168.67.1
server 192.168.34.1
server 192.168.34.2
peer 192.168.67.3


; NTP Configuration for Computer Room Host 192.168.67.x
server 192.168.67.1
server 192.168.67.2
server 192.168.67.3
peer 192.168.67.y
peer 192.168.67.z
.
.
.
```

Troubleshooting

NTP provides error messages for troubleshooting.

Note! You can also use NTPQ to troubleshoot NTP.

Troubleshooting Tips

Here are some troubleshooting tips:

- Make sure the entries in the NTP configuration file TCPWARE:NTP.CONF are correct. At the minimum, there must be a server or peer declaration for a machine that is reachable, and if authentication is enabled, set it up to properly authenticate NTP packets. This machine serving time must be connected either to lower stratum machines or to some reference time source.

- Make sure that the logical TCPWARE_TIMEZONE is properly defined to reflect the timezone (and daylight savings). If the logical is undefined or incorrect, NTP is likely to abort. TCPWARE_TIMEZONE should be set by configuring TCPware with the configuration procedure TCPWARE:CNFNET.COM.
- If using the **`slewalways`** command, make sure the system time is within 4000 seconds of the correct time before starting NTPD. If the local system time is off by more than 4000 seconds from server time, NTPD logs a message and stops running. Also, if the local clock is not within a minute or two of correct time when starting NTPD with slewalways set, it may take some time for NTPD to synchronize the clock. Ideally, set the clock with NTPDATE or SET TIME before starting NTPD.
- Make sure that TIMED and DTSS services are not running on the system. These services are used to synchronize time, and interfere with NTP unless NTP was configured in special cases to work with them. (See the **`master-clock`** command.)

The following messages are generated by the the NTP server. They go to both OPCOM and the TCPWARE:NTPSERVER.LOG file. This log file is the best source of information for troubleshooting in that it contains a record of these messages as well as additional informational messages. Messages appear in the log file without the bracketed prefix.

There are four types of messages generated:

- Configuration messages
- Peer contact messages
- Synchronization messages
- Unexpected error condition messages

Access error messages help by entering **`HELP TCPWARE MESSAGES.`**

Troubleshooting Using NTPQ

The NTPQ utility has a few commands that are helpful in identifying problems. The peers command is one of the simplest and is a quick way to check the offset (time difference) between the local host and peer machines.

The `readvar` command is useful for more in depth information. Without arguments, it displays information about the local host. When `readvar` is followed by an *assocID*, it displays information about the peer corresponding to the *assocID* (use `associations` to display the *assocIDs* for all peers). Of interest is the record of time offsets and round trip delays for packets (the `filtoffset` and `filtdelay` fields). This provides a record of the last eight time updates obtained from a peer.

The command `readvar` *assocID* flash displays a useful variable, `flash`, which can be of particular interest for troubleshooting. The bits in the `flash` variable, if set, have the following meaning in relation to a peer:

```
0x01    /* duplicate packet received */
0x02    /* bogus packet received */
```

```
0x04    /* protocol unsynchronized */
0x08    /* peer delay/dispersion bounds check */
0x10    /* peer authentication failed */
0x20    /* peer clock unsynchronized */
0x40    /* peer stratum out of bounds */
0x80    /* root delay/dispersion bounds check */
```

NTPQ, XNTPDC, NTPDATE, and NTPTRACE Utilities

The following sections describe the NTPQ, XNTPDC, NTPDATE, and NTPTRACE utilities.

ntpq

The NTPQ utility is used to query NTP servers that implement the recommended NTP mode 6 control message format about current state and to request changes in that state. The program runs interactively or uses command line arguments. Requests to read and write arbitrary variables can be assembled, with output options available. NTPQ can also obtain and print a list of peers in a common format by sending multiple queries to the server.

The utility uses NTP mode 6 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network which permits it. Note that since NTP is a UDP protocol, this communication is somewhat unreliable, especially over large distances in terms of network topology. NTPQ makes one attempt to retransmit requests, and times out requests if the remote host is not heard from within a suitable timeout time.

Interactive Commands

Internal Commands

Interactive format commands consist of a keyword followed by zero to four arguments. Enter only the minimum number of characters of the full keyword to uniquely identify the command. The output is normally sent to the standard output, but you can send the output to a file by appending a greater than (>) followed by a filename to the command line.

? ***[command-keyword]***
help ***[command-keyword]***

A question mark **`(?)`** by itself prints a list of all the known command keywords. A question mark followed by a command keyword prints function and usage information.

cooked

Causes output from query commands to be "cooked" for user readability. Variables the server recognizes have their values reformatted for readability. Variables that NTPQ determines should have a decodeable value, but do not, are marked with a trailing question mark **`(?)`**.

debug

Enables debugging. The format is

```
debug [no|more|less]
```

where:

no	turns off debug.
more	increments debug level by 1.
less	decrements debug level by 1.

If no value is specified, displays the current debug level.

exit

Exits from NTPQ.

host *[hostname]*

Sets the host to which future queries are sent. Hostname may be either a hostname or a numeric address. If *hostname* is omitted, the host currently set is displayed.

hostnames *[* yes | no *]*

If specifying **`hostnames yes`** (the default), hostnames are printed in information displays. If specifying **`hostnames no`**, numeric addresses are printed instead.

ntpversion *[* 1 | 2 | 3 *]*

Sets the NTP version number that NTPQ claims in packets. Note that modes did not exist in NTP version 1. (There appears to be no servers left that demand version 1.) If the version value is omitted, the version number (default 3) currently in use is displayed.

passwd

Specifies a password for authenticated requests. The format is

```
passwd password
```

quit

Exits NTPQ.

raw

Causes all output from query commands to be printed as received from the remote server. The only formatting or interpretation done on the data is to transform non-ASCII data into a printable form.

timeout *[milliseconds]*

Specifies a timeout period for responses to server queries. The default is about 5000 milliseconds. Note that since NTPQ retries each query once after a timeout, the total waiting time for a timeout is twice the timeout value set. If the *milliseconds* value is omitted, the current timeout period is displayed.

version

Prints the version of this program.

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 to which the values correspond by including its association ID. An association ID of 0 is special, and indicates the variables are system variables whose names are drawn from a separate name space.

addvars

Adds variables to the variable list. The format is

```
addvars name[=value][,...]
```

name	is the variable name to add.
value	is the value to assign to name.

authenticate

Toggles whether requests are authenticated. The format is

```
authenticate (yes|no)
```

If neither yes/no is specified, the current authentication status is displayed.

associations

Obtains and prints a list of association identifiers and peer status for in-spec peers of the server being queried. Each line in the list is associated with the corresponding peer line given by the **`peers`** command. 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 the actual association identifier returned by the server, and the third is the status word for the peer, as described in Appendix B.2.2 of the NTP specification RFC 1305. This is followed by a number of columns containing data decoded from the status word.

cl

Reads the clock variables included in the variable list. The format is

```
cl [associd]
```

associd	is the association ID of the system to read. A 0 or a missing value indicates to use the local system.

clearvars

Removes all variables from the variable list

clocklist

Reads the clock variables from the specified system. The format is

```
clocklist [associd]
```

associd	is the association ID of the system to read. A 0 or a missing value indicates to use the local system.

clockvar

Requests that a list of the server's clock variables be sent. Servers that have a radio clock or other external synchronization respond in the affirmative to this. For a TCPware NTP server the request refers to the local clock, if configured for NTPD by **`master-clock`** or **`local-master`**.

cv

Reads the clock variables from the system specified by associd. The format is

```
cv [associd] [name=value[,...]]
```

associd	is the association for the system to read. If associd is not specified, 0 (the local system) is used.
name	is the variable name to read.

delay

Sets the delay (in milliseconds) added to encryption time stamps. The format is

```
delay [msec]
```

msec	delays to add to the authenticated requests. Not specifying a delay value causes ntpq to display the current delay value.

keyid

Sets the keyid to use for authenticated requests. The format is

```
keyid [keyid]
```

keyid	is a 32-bit integer to use as the key. Not specifying a key value will cause ntpq to display the current keyid.

keytype

Sets the key type to use for authenticated requests. The format is

```
keytype type
```

type	is either des or md5.

lassociations

Prints the list of associations including all client information.

lopeers

Obtains and prints a list of all peers and clients with full information.

lpassociations

Prints the last obtained list of associations.

lpeers

Obtains and prints a list of all peers and clients.

mreadlist/mrl

Reads peer variables in the variable list for multiple peers specified by the list of associd's.

```
mreadlist assocID [assocID...]
```

associd	is the association ID of the system to read. A 0 or a missing value indicates to use the local system.

mreadvar/mrv

Reads the peer variables from the multiple peers specified by the list of associd's and variables. The format is

```
readvar [associd] [name=value[,...]]
```

associd	is the association ID of the system to read. A 0 or a missing value indicates to use the local system.

name	is the variable name to read. Not specifying a name will cause all variables to be read.

ntpversion

Sets the NTP version number to use for request packets. The format is

```
ntpversion (1|2|3)
```

opeers

Prints the peer list with dest addr shown instead of REFID.

passociations

Prints the list of associations returned by last associations command.

peers

Obtains a list of in-spec 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 ref ID is unknown); stratum of the remote peer; type of the peer (local, unicast, multicast, or broadcast), when the last packet was received; polling interval (in seconds); reachability register (in octal); and the current estimated delay, offset and dispersion of the peer (all in milliseconds).

The character in the left margin indicates the fate of this peer in the clock selection process:

<space>	Discarded due to high stratum and/or failed sanity checks
x	Falseticker
.	Culled from the end of the candidate list
-	Discarded by the clustering algorithm
+	Included in the final selection set
#	Selected for synchronization but the distance exceeds the maximum
*****	Selected for synchronization
o	Selected for synchronization, a PPS signal is in use

Note! Since the **`peers`** command depends on the ability to parse the values in the responses it gets, it may fail to work with servers that poorly control the data formats.

The contents of the host field may be in one of four forms. It may be a hostname, an IP address, a reference clock implementation name with its parameter or `REFCLK(, )`. With **`hostnames no`**, only IP addresses are displayed.

poll

Polls an NTP server in client mode *n* times.

pstatus ***assocID***

Sends a read status request to the server for the given association. (See the **associations** command for *assocIDs*). The names and values of the peer variables returned are printed. Note that the status word from the header is displayed preceding the variables, both in hexidecimal and in English.

readlist/rl

Reads the system or peer variables from the system specified by associd. The format is

```
readlist [associd]
```

associd	is the association ID of the system to read. A 0 or a missing value indicates to use the local system.

readvar ***[assocID [variable-name[=value] [...]]]***
rv ***[assocID [variable-name[=value] [...]]]***

Requests that the values of the specified variables be returned by the server, by sending a read variables request. If you omit the association ID or give it as zero, the variables are system variables; otherwise they are peer variables and the values returned are those of the corresponding peer. (See the **associations** command for *assocIDs*). Omitting the variable list sends a request with no data, which should induce the server to return a default display. If more than one variable is requested, separate the variable list with commas and do not include spaces.

rmvars

Removes variables from the variable list. The format is

```
rmvars name[,...]
```

name	is the variable to remove from the variable list.

showvars

Prints the variables on the variable list.

writelist

Writes the system or peer variable included in the list. The format is

```
writelist [associd]
```

associd	is the association ID of the system to write. A 0 or a missing value indicates to use the local system.

writevar

Writes the system or peer variables. The format is

```
writevar associd name=value,[...]
```

associd	is the association ID of the system to write. A 0 or a missing value indicates to use the local system.
name	is the variable name to write.

Command Line Format

ntpq ***[* -n *] [-c command] [host] [...]***

(If command line arguments are omitted, NTPQ runs in interactive mode.)

-c

The *command* that follows is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s). The *command* must be in double quotes if it consists of more than one word. Multiple **`-c`** options can be given.

-d

Enables debugging within NTPQ.

-i

Enters interactive mode. This is the same as typing “ntpq” with no switches.

-n

Displays all host addresses in dotted quad numeric format rather than converting them to canonical hostnames.

-p

Lists the peers for this host.

host

Sets the host to which future queries are sent, as either a hostname or a numeric address. If *host* is omitted, the local host is used.

xntpdc

The XNTPDC utility is used to query the NTPD daemon about its current state and to request changes in that state. The program runs interactively or uses command line arguments. Extensive state and statistics information is available through the XNTPDC interface. In addition, nearly all the configuration options that can be specified at startup using NTPD’s configuration file may also be specified at run-time using XNTPDC.

The XNTPDC utility uses NTP mode 7 packets to communicate with the NTP server, and can be used to query any compatible server on the network which permits it. Note that since NTP is a UDP protocol, this communication is somewhat unreliable, especially over large distances, in terms of network topology. XNTPDC makes no attempt to retransmit requests, and times out requests if the remote host is not heard from within a suitable timeout time.

XNTPDC's operation is specific to the NTPD implementation and can be expected to work only with this, and possibly some previous versions, of the daemon. Requests from a remote XNTPDC program that affect the state of the local server must be authenticated, which requires both the remote program and local server to share a common key and key identifier.

Interactive Commands

Internal Commands

Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. The output of a command is normally sent to the standard output, but you can send the output of individual commands to a file by appending a greater than (>) followed by a filename to the command line.

? ***[command-keyword]***
help ***[command-keyword]***

A question mark **`(?)`** by itself prints a list of all the known command keywords. A question mark **`(?)`** followed by a command keyword prints function and usage information.

debug

Enables debugging. The format is

```
debug [no|more|less]
```

no	turns off debug.
more	increments debug level by 1.
less	decrements debug level by 1.

If no value is specified, display the current debug level.

delay ***milliseconds***

Specifies a time interval to be added to timestamps included in requests that require authentication. This is used to enable unreliable server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized.

exit

Exits from the program.

host *[hostname]*

Sets the host to which future queries are sent. *Hostname* may be either a hostname or a numeric address. If the *hostname* is omitted, the host currently set is displayed.

hostnames *[* yes | no *]*

If specified as **`hostnames yes`** (the default), hostnames are printed in information displays. If specified as **`hostnames no`**, numeric addresses are printed instead.

keyid *[keyid]*

Allows a key number to be used by XNTPDC to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose. If the *keyid* is omitted, the keyid currently set is displayed.

keytype

Sets the key type to use for authenticated requests. The format is

```
keytype type
```

type	is either des or md5.

quit

Exits NTPQ.

passwd

Prompts you to type in a password (which is not echoed) that is 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. If the **`keyid`** command was not used to set the keyid, you are prompted for the keyid.

timeout *[milliseconds]*

Specifies a timeout period for responses to server queries. The default is approximately 8000 milliseconds. Note that since XNTPDC retries each query once after a timeout, the total waiting time for a timeout is twice the timeout value set. If the *milliseconds* value is omitted, the current timeout period is displayed.

version

Displays the version number of xntpdc.

Control Message Commands

Query commands produce NTP mode 7 packets containing requests for information being sent to the server. These are read-only commands in that they make no modification of the server configuration state.

addrefclock

Configures a new server. The format is

```
addrefclock addr [mode] [minpolls|prefer] [minpoll|prefer]
```

addr	the address of the server.
mode	values are 1 for symmetric active server or 2 for symmetric passive server.
minpoll	the minimum time (in seconds) between polls. The range is 4-14 seconds; the default is 4 seconds.
prefer	the preferred server (default not preferred).

In this example:

```
addrefclock 207.225.29.52 1 10 prefer
```

adds system 207.225.29.52 as a symmetric active server with 10 seconds between polls, and this is a preferred server.

addtrap

Sets the server trap for asynchronous messages from a system. The format is

```
addtrap address [port] [interface]
```

address	is the address of the orignation system to trap.
port	is the TCP/IP port to watch.
interface	is the interface to watch.

clkbug

Displays clock debugging information. This is used for user-written reference clock drivers.

clockstat

Displays clock status information from a peer. The format is

```
clockstat address [address1 address2 address3]
```

address	is the address of the peer.

clrtrap

Clears traps in the server. The format is

```
addtrap address [port] [interface]
```

address	is the address of the orignation system to trap.
port	is the TCP/IP port to watch.
interface	is the interface to watch.

controlkey

Changes the keyid the server uses to authenticate control messages. The format is

```
controlkey keyid
```

keyid	is the unsigned 32-bit integer to use.

ctlstats

Displays the packet count statistics from the control module. The format is

```
ctlstats
```

dmpeers

Displays peer summary information.

fudge

Sets/changes one of a clock's fudge factors. The settings are dependant upon the reference clock driver being used.

iostats

Prints statistics counters maintained in the input-output module.

kerninfo

Returns kernel information (not supported for VMS hosts).

leapinfo

Displays the current leap second state.

listpeers

Obtains and prints a brief list of the peers for which the server is maintaining state. These should 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 synchronization candidates.

loopinfo *[* oneline | multiline *]*

Prints the values of selected loop filter variables:

`loop filter`	The part of NTP that deals with adjusting the local system clock
`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 (ppm)
`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 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 printed, with **`multiline`** as the default.

memstats

Prints statistics counters related to memory allocation code.

monlist *[version]*

Obtains and prints traffic counts collected and maintained by the monitor facility. You do not normally need to specify the version number.

peers

Obtains a list of peers for which the server is maintaining state, along with a summary of that state. Summary information includes the address of the remote peer; local interface address (0.0.0.0 if a local address has yet to be determined); stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized); polling interval (in seconds); reachability register (in octal); and current estimated delay, offset, and dispersion of the peer (all in seconds).

In addition, the character in the left margin indicates the mode this peer entry is operating in:

+	Symmetric active	**^**	Server is broadcasting to this address
-	Symmetric passive	**~**	Remote peer is sending broadcasts
=	Remote server is being polled in client mode	*****	Peer the server is currently synchronizing to

The contents of the host field may be in one of four forms: a hostname, IP address, reference clock implementation name with its parameter, or **`REFCLK`** (implementation number, parameter). With **`hostnames no`**, only IP-addresses are displayed.

preset

Resets stat counters associated with particular peers.

pstats *peer-address [...]*

Shows per-peer statistic counters associated with the specified peers.

requestkey

Changes the keyid the server uses to authenticate requests. The format is

```
requestkey keyid
```

keyid	is the 32-bit integer to use as the ID.

reslist

Obtains and prints the server's restriction list. This list is usually printed in sorted order and may help to understand how the restrictions are applied.

showpeer *peer-address [...]*

Shows a detailed display of the current peer variables for one or more peers. Most of these values are described in the NTP Version 2 specification.

sysinfo

Prints a variety of system state variables, such as the state related to the local server. All except the last four lines are described in the NTP Version 3 specification, RFC-1305.

The system flags can be set and cleared by the **`enable`** and **`disable`** configuration commands, respectively. These are the `auth, bclient, monitor, pll, pps,` and `stats` flags. (See the NTPD utility for the meaning of these flags.)

The stability is the residual frequency error remaining after the system frequency correction is applied, and is intended for maintenance and debugging. In most architectures, this value initially decreases from as high as 500 ppm to a nominal value in the range .01 to 0.1 ppm. If it remains high for some time after starting the daemon, something might be wrong with the local clock.

The `broadcastdelay` shows the default broadcast delay. The `authdelay` shows the default authentication delay, as computed by NTPD for authorization.

sysstats

Prints statistics counters maintained in the protocol module.

timerstats

Prints statistics counters maintained in the timer/event queue support code.

traps

Displays the traps set in the server.

Runtime Configuration Requests

All requests that cause state changes in the server are authenticated by the server using the

`requestkey` in the configuration file (which can be disabled by the server by not configuring a key). The key number and the corresponding key must also be made known to XTNPDC. This can be done using XNTPDC's **`keyid`** and **`passwd`** commands, the latter of which prompts at the terminal for a password to use as the encryption key. You are also prompted automatically for both the key number and password the first time a command is given that would result in an authenticated request to the server. Authentication not only provides verification that the requester has permission to make such changes, but also gives an extra degree of protection against transmission errors.

Authenticated requests always include a timestamp in the packet data, which is included in the computation of the authentication code. This timestamp is compared by the server to its receive timestamp. If they differ by more than a small amount, the request is rejected. This is done for two reasons. First, it makes simple replay attacks on the server, by someone who might be able to overhear traffic on your LAN, much more difficult. Secondly, it makes it more difficult to request configuration changes to your server from topologically remote hosts. While the reconfiguration facility works well with a server on the local host, and may work adequately between time synchronized hosts on the same LAN, it works very poorly for more distant hosts. As such, if reasonable passwords are chosen, care is taken in the distribution and protection of keys, and appropriate source address restrictions are applied, the run-time reconfiguration facility should provide an adequate level of security.

The following commands all make authenticated requests.

addpeer ***peer-address [keyid] [version] [prefer]***

Adds a configured peer association at the given address and operates in symmetric active mode. Note that an existing association with the same peer may be deleted when this command is executed, or may simply be converted to conform to the new configuration, as appropriate. If the optional *keyid* is a nonzero integer, all outgoing packets to the remote server have an authentication field attached, encrypted with this key. If the value is 0 (or not given), no authentication is done. The *version* can be 1, 2, or 3, and defaults to 3. The **`prefer`** keyword indicates a preferred peer (and thus is used primarily for clock synchronization if possible).

addserver ***peer-address [keyid] [version] [prefer]***

Identical to the **`addpeer`** command, except that the operating mode is client.

broadcast ***peer-address [keyid] [version] [prefer]***

Identical to the **`addpeer`** command, except that the operating mode is broadcast. In this case a valid key identifier and key are required. The *peer-address* parameter can be the broadcast address of the local network, or a multicast group address assigned to NTP. If using a multicast address, a multicast-capable kernel is required.

unconfig ***peer-address [...]***

Removes the configured bit from the specified peers. In many cases, 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] [...]*

Operates the same as the `enable` and `disable` configuration file commands of NTPD.

restrict *address mask flag [flag]*

Operates the same as the restrict configuration file commands of NTPD.

unrestrict *address mask flag [flag]*

Unrestricts the matching entry from the restrict list.

delrestrict *address mask [ntpport]*

Deletes the matching entry from the restrict list.

readkeys

Causes the current set of authentication keys to be purged and a new set to be obtained by rereading the keys file (TCPWARE:NTP.KEYS). This allows encryption keys to be changed without restarting the server.

trustedkey *keyid [...]*
untrustedkey *keyid [...]*

Operates the same as the trustedkey and untrustedkey configuration file commands of NTPD.

authinfo

Returns information concerning the authentication module, including known keys and counts of encryptions and decryptions which have been done.

reset

Clears the statistics counters in various modules of the server.

Command Line Format

xntpdc *[-n] [-c command] [host] [...]*

Command Line Arguments

(If command line arguments are omitted, XNTPDC runs in interactive mode.)

-c

The *command* that follows is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s). The *command* must be in double quotes if it consists of more than one word. Multiple **`-c`** options can be given.

-d

Enables debugging within XNTPDC.

-i

Enters interactive mode. This is the same as typing “xntpdc” with no switches.

-n

Displays all host addresses in dotted quad numeric format rather than converting them to canonical hostnames.

-p

Lists the peers within XNTPDC.

host

Sets the host to which future queries are sent, as either a hostname or a numeric address. If *host* is omitted, the local host is used.

-l

Shows the peers for this host, showing only the peer name and no other status information.

-s

This is the same information as -p, except that the first column of the display (*, +, ., -) is not displayed.

ntpdate

The NTPDATE utility sets the local date and time, by polling the NTP servers given as the server arguments, 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 of these. Note that the accuracy and reliability of ntpdate depends on the number of servers, the number of polls each time it is run, and the interval between runs.

The NTPDATE utility can be run manually as necessary to set the host clock, or it can be run from the system startup command file to set the clock at boot time. This is useful in some cases to set the clock initially before starting the NTP daemon, NTPD. It is also possible to run NTPDATE from a batch job. However, it is important to note that NTPDATE with contrived batch jobs is no substitute for the NTP daemon, which uses sophisticated algorithms to maximize accuracy and reliability while minimizing resource use. Finally, since NTPDATE does not discipline the host clock frequency as does NTPD, the accuracy using NTPDATE is limited.

The NTPDATE utility makes time adjustments in one of two ways. If it determines that the clock is wrong by more than 0.5 second, it simply steps the time by calling the $SETIME system service. If the error is less than 0.5 seconds, it slews the time by temporarily adjusting system clock variables. The latter technique is less disruptive and more accurate when the error is small, and works quite well when NTPDATE is run by a batch job every hour or two.

The NTPDATE utility declines to set the date if NTPD is running on the same host. When running NTPDATE every hour or two from a batch job, as an alternative to running NTPD, results in precise enough timekeeping to avoid stepping the clock.

Format

ntpdate *[-bdqsv] [-a key#] [-e delay] [-k file] [-p samples] [-o version#] [-t timeo] [server]*

Command Line Options

-a *key*

Enables the authentication function and specifies the key identifier to be used for authentication as the argument key. The keys and key identifiers must match in both the client and server key files. The default is to disable the authentication function.

-B

Forces the time to always be slewed, even if the measured offset is greater than "128 ms. The default is to step the time if the offset is greater than "128 ms. Note that, if the offset is much greater than "128 ms in this case, that it can take several hours to slew the clock to the correct value. During this time, the host should not be used to synchronize clients.

-b

Forces the time to be stepped, rather than slewed (default). This option should be used when called from a startup file at boot time.

-d

Enables debug mode.

-e

Sets the encryption delay to *nnn*.

-k *keyfile*

Specifies the path for the authentication key file as the string keyfile. The default is TCPWARE:NTP.KEYS. This file should be in the format described for NTPD configuration.

-o *version*

Specifies the NTP version for outgoing packets as the integer version, which can be 1 or 2. The default is 3. This allows NTPDATE to be used with older NTP versions.

-p *samples*

Specifies the number of samples to be acquired from each server as the integer samples, with values from 1 to 8 inclusive. The default is 4.

-q

Uses simple query mode (do not attempt to bind the NTP socket).

-s

Enables OPCOM messaging. This is designed primarily for the convenience of batch jobs.

-t *timeout*

Specifies the maximum time waiting for a server response as the value timeout, in seconds and fraction. The value is is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

-u

Directs NTPDATE to use an unprivileged port on outgoing packets. This is most useful when behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronize with hosts beyond the firewall.

-v

Sets verbose mode.

ntptrace

The NTPTRACE utility determines where a given NTP server gets its time from, and follows the chain of NTP servers back to their master time source. If given no arguments, it starts with localhost. Here is an example of the output from NTPTRACE:

```
$ ntptrace
localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2ozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid
'WWVB'
```

On each line, the fields are (left to right): the host name, host stratum, time offset between that host and the local host (as measured by NTPTRACE; this is why it is not always zero for localhost), host synchronization distance, and (only for stratum-1 servers) the reference clock ID. All times are given in seconds. Note that the stratum is the server hop count to the primary source, while the synchronization distance is the estimated error relative to the primary source. The NTP server must be synchronized to a peer.

Format

ntptrace *[-dnv] [-o version#] [-r retries] [* -t *timeout] [server]*

Command Line Options

-d

Enables debug mode.

-n

Turns off the printing of hostnames; instead, host IP addresses are given. This may be useful if a nameserver is down.

-o

Sets the protocol version to be used for the packets sent.

-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

Prints verbose information about the NTP servers.

Chapter 10

TIMED

Introduction

This chapter describes how to manage TIMED, the time daemon, otherwise known as the Time Synchronization Protocol (TSP). TIMED synchronizes the clocks of the various hosts in a LAN. The synchronization process is totally transparent to system or network users.

The `timed` program (or daemon) uses the Time Synchronization Protocol (TSP) to synchronize the clocks of LAN hosts. TSP was originally based on UNIX 4.3BSD systems, but is available for all systems connected by TCPware for OpenVMS.

TIMED provides:

- Synchronization of host clocks
- Election of primary candidates if the primary disappears

Time Synchronization

TIMED synchronizes clocks through an algorithm that slows some clocks and speeds up others to create an average network time. Each host must support a time daemon. The `timed` daemon depends on a primary/dependent relationship. The primary time daemon measures the difference between clocks using broadcast ICMP timestamp requests and computes the network time based on the average of all replies. It then sends the dependent time daemons the corrections they should make to their clocks. Synchronization can occur to within 20 milliseconds.

This process repeats periodically. If host clocks drift away from each other, the process again synchronizes them based on the average. Because corrections depend on time differences rather than absolute time, transmission delays have no effect on synchronization.

When a new dependent host joins the network, its time daemon immediately resets its clock based on that of the primary time server. In the case of a fixed primary, no averaging takes place and each negotiating dependent accepts the absolute time of the primary host.

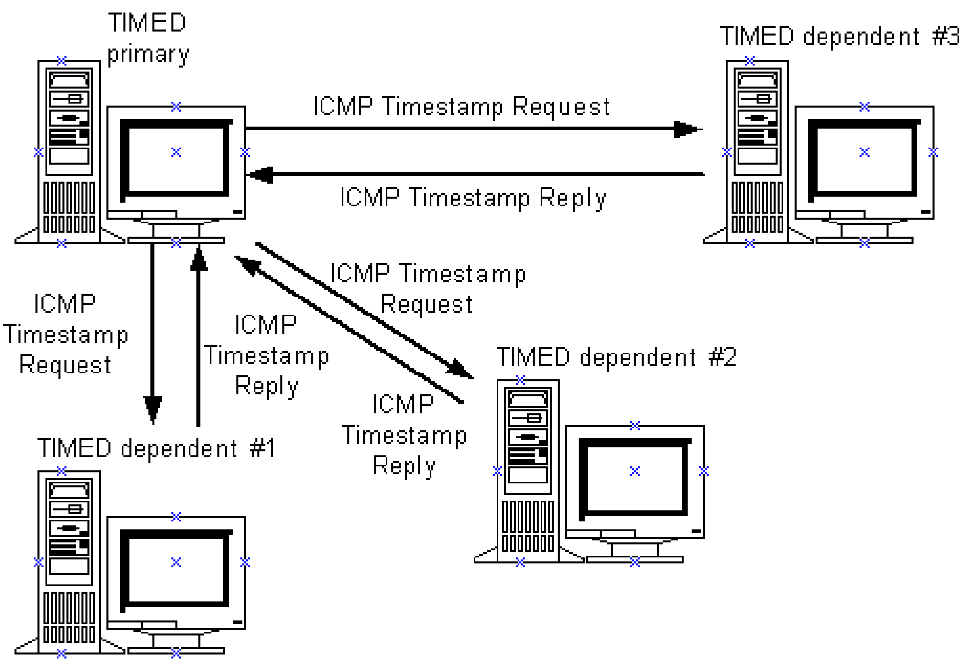
The `timed` daemon interacts with the `date` command to set the date globally. You can also use the

TIMEDC command to determine the status of a time daemon (see *TIMEDC Command Reference*).

Figure 10-1 shows a typical time synchronization transaction, as follows:

1	When a dependent host initially starts TIMED, it requests the network time from the primary host and adjusts its time accordingly.
2	The primary broadcasts messages to solicit the names of active dependents, which reply.
3	The primary sends an "adjust time" message to each dependent with the difference between the dependent timestamp and the computed average time.
4	Each dependent sends back an acknowledgment.

Figure 10-1 Time Synchronization Transaction Using TSP



Primary Candidate Election Process

Dependent hosts also need to elect among themselves candidates for a new primary should the current one go down. This occurs through an election algorithm, as follows:

1	When a dependent's election timer expires, it broadcasts an "election" message declaring that it wants to become the new primary. The dependent becomes a primary candidate.

2	If no other dependent was accepted as primary candidate, each of the other dependents responds with an "accept" message to the election request and rejects any other candidates.
3	The primary candidate adds each accepting dependent's name to its list.
4	If the current primary goes down, the elected candidate becomes the new primary.

TIMED Operation Mode

You can start TIMED in one of the following three modes:

Mode	**Description**
Dependent	The host joins the time synchronization if it finds a primary daemon on the network. It never operates as a primary.
Primary candidate	If the host finds a primary daemon on the network, it operates as a dependent daemon. If it does not find a primary daemon, it initiates a primary election process, and if granted, becomes a primary if the primary server goes down. There can be multiple TIMED hosts started in primary candidate mode.
Fixed primary	The host starts as a primary server or receives an error if it detects another primary. A fixed primary adjusts other hosts to its host time. It does not average the network time and never adjusts its own time. All other hosts must run as dependents.

You can synchronize the clock of the host running a fixed primary with other reliable sources using different clock synchronization. For example, you can run the Network Time Protocol (NTP) daemon to synchronize the fixed primary host to reliable sources on the Internet.

See Chapter 9, *Network Time Protocol (NTP)*, for details on NTP.

Changing Network Time

When TIMED synchronizes the clocks on multiple hosts, changing time on one machine using the OpenVMS SET TIME command does not achieve the desired effect unless the host for which you change the time is a fixed primary.

To adjust the time of all hosts synchronized by TIMED, use the DATE option in the TIMEDC command (see *TIMEDC Command Reference*). If there are UNIX hosts synchronized with TIMED, the UNIX `date` command adjusts the network time.

Setting TIMED Parameters

The `timed` daemon starts up automatically enabled during TCPware configuration. You can set certain parameters on startup through TCPware logicals. See the following table:

Logical	Description
TCPWARE_TIMED_MODE	Determines if the current host is a MASTER (primary), FIXED MASTER (fixed primary), or SLAVE (dependent) **MASTER** (primary)--broadcasts time synchronization requests, calculates the time differences and averages, and sends "adjust time" messages. **FIXED MASTER** (fixed primary)--provides absolute time stamps to newly started dependent TIMED hosts. **SLAVE** (dependent)--is the recipient of primary "adjust time" messages.
TCPWARE_TIMED_INCLUDE	Determines the networks included in clock synchronization, either in network addresses or names
TCPWARE_TIMED_EXCLUDE	Determines the networks excluded from clock synchronization, either in network addresses or names

Loadable Timezone Rules

Loadable timezone rules provided with TCPware are in the text file TCPWARE:TIMEZONES.DAT. You can add user-written timezone rules to the file TCPWARE:TIMEZONES.LOCAL to override the zones in TIMEZONES.DAT. Loadable timezone rules consist of three parts:

COUNTRY	Is a collection of timezones (ZONEs). For example, the country US selects all U.S. timezones. This provides a convenient way to select groups of timezones.
ZONE	Is a specification of a particular timezone, including the name of the zone, the GMT offset, the DST rules in effect, and the name to use while DST is in effect.
RULE	Is a rule for determining when DST is in effect.

Format of COUNTRY Specification

```
COUNTRY countryname zonename [zonename . . .]
```

The COUNTRY specification gives the name of a geographical area and the names of the timezones associated with it. This provides a way to group timezones so they may be selected more conveniently.

The following example shows the definition of the country "US" listing the zones corresponding to the United States. The example for Arizona is slightly different, showing the zone "US/Arizona" instead of "US/Mountain." ("US/Arizona" is the definition of a Mountain timezone that does not observe Daylight Savings Time.)

```
Country US  US/Eastern US/Central US/Mountain US/Pacific US/Yukon
US/Hawaii

Country US/Arizona -

     US/Eastern US/Central US/Arizona US/Pacific US/Yukon US/Hawaii
```

Format of ZONE Specification

ZONE *zonename gmtoffset rulename standard-name dst-name* [COMPILED_IN]

zonename	Is the name by which this zone can be selected, or the name by which it is referred to in a COUNTRY specification.
gmtoffset	Is this zone’s standard time offset from GMT.
rulename	Is the name of the RULE specification that determines when DST is in effect for this zone. The rulename may be an underscore (_) to indicate that this zone does not use DST.
standard-name and *dst-name*	Are the names by which this zone is referred to during standard time, and during Daylight Savings Time, respectively. These are the names by which SET TIMEZONE selects the local timezone.

The ZONE specification describes a timezone:

If there are no DST rules, the dst-name should be specified as an underscore (_). The optional COMPILED_IN keyword indicates that this rule is compiled-in and need not be loaded, as long as no other rules conflict with it. If you edit a COMPILED_IN ZONE specification, you must remove the COMPILED_IN keyword to force the ZONE specification to be loaded.

The following example shows the definition of the normal United States Mountain timezone. The Arizona example shows the definition of a Mountain timezone that does not observe Daylight Savings Time.

```
Zone    US/Mountain     -7:00   US  MST  MDT COMPILED_IN
Zone    US/Arizona      -7:00   _   MST
```

Format of RULE Specification

RULE *rulename startyear ruletype save start-date end-date*

The RULE specification describes a set of rules for determining at what times DST is in effect:

rulename	Is the name of the RULE specification in ZONE specifications.

startyear	Is the year during which this DST rule takes effect. The rule remains in effect until a later *startyear* is specified in a rule with the name *rulename*.
ruletype	Specifies the type of DST rules. There are three permitted values: • DST indicates normal Northern-Hemisphere Daylight Savings Time rules, which switch at the time and date indicated. • REV_DST indicates normal Southern-Hemisphere Daylight Savings Time rules. NULL indicates that no Daylight Savings Time is in effect during the specified years.
save	Indicates the difference between Standard Time and DST.
start-date and *end-date*	Specify the starting and ending dates for DST. Specific dates can be specified, or rules such as "First Sunday" or "Last Sunday" can be used. See the file TCPWARE:TIMEZONES.DAT for examples on specifying dates.

The following example illustrates the United States

Federal Daylight Savings Time rules:

```
Rule US 1987 DST 1:00  First Sunday April 2:00 Last Sunday October 2:00
Rule US 1976 DST 1:00  Last  Sunday April 2:00 Last Sunday October 2:00
Rule US 1975 DST 1:00  23    February     2:00 Last Sunday October 2:00
Rule US 1974 DST 1:00  6     January      2:00 Last Sunday October 2:00
Rule US 1970 DST 1:00  Last  Sunday April 2:00 Last Sunday October 2:00
```

Loadable Timezone Rules Provided with TCPware

Table 7-2 shows the loadable rules provided in the TCPWARE:TIMEZONES.DAT file which you may modify or augment as appropriate for your location.

Table 10-1   Loadable Timezone Rules

Country Name	Rule Name	Timezone Name	GMT Offset	DST Rules
	GMT	GMT[a]	0 hours	-none-
	UT	UT[a]	0 hours	-none-
US-Military	US-Military/Z[a]	Z	0 hours	-none-
US-Military	US-Military/A[a]	A	-1 hour	-none-
US-Military	US-Military/B[a]	B	-2 hours	-none-

Table 10-1 Loadable Timezone Rules (Continued)

Country Name	Rule Name	Timezone Name	GMT Offset	DST Rules
US-Military	US-Military/C[a]	C	-3 hours	-none-
US-Military	US-Military/D[a]	D	-4 hours	-none-
US-Military	US-Military/E[a]	E	-5 hours	-none-
US-Military	US-Military/F[a]	F	-6 hours	-none-
US-Military	US-Military/G[a]	G	-7 hours	-none-
US-Military	US-Military/H[a]	H	-8 hours	-none-
US-Military	US-Military/I[a]	I	-9 hours	-none-
US-Military	US-Military/K[a]	K	-10 hours	-none-
US-Military	US-Military/L[a]	L	-11 hours	-none-
US-Military	US-Military/M[a]	M	-12 hours	-none-
US-Military	US-Military/N[a]	N	1 hour	-none-
US-Military	US-Military/O[a]	O	2 hours	-none-
US-Military	US-Military/P[a]	P	3 hours	-none-
US-Military	US-Military/Q[a]	Q	4 hours	-none-
US-Military	US-Military/R[a]	R	5 hours	-none-
US-Military	US-Military/S[a]	S	6 hours	-none-
US-Military	US-Military/T[a]	T	7 hours	-none-
US-Military	US-Military/U[a]	U	8 hours	-none-
US-Military	US-Military/V[a]	V	9 hours	-none-
US-Military	US-Military/W[a]	W	10 hours	-none-
US-Military	US-Military/X[a]	X	11 hours	-none-
US-Military	US-Military/Y[a]	Y	12 hours	-none-

Table 10-1 Loadable Timezone Rules (Continued)

Country Name	Rule Name	Timezone Name	GMT Offset	DST Rules
US	US/Eastern[a]	EST/EDT	-5 hours	US Federal
US	US/Central[a]	CST/CDT	-6 hours	US Federal
US	US/Mountain[a]	MST/MDT	-7 hours	US Federal
US	US/Pacific[a]	PST/PDT	-8 hours	US Federal
US	US/Yukon[a]	YST/YDT	-9 hours	US Federal
US	US/Hawaii[a]	HST	-10 hours	-none-
US/East-Indiana	US/East-Indiana	EST	-5 hours	-none-
US/Arizona	US/Arizona	MST	-7 hours	-none-
Canada	Canada/ Newfoundland[a]	NST/NDT	-3:30 hours	Canadian
Canada	Canada/Atlantic[a]	AST/ADT	-4 hours	Canadian
Canada	Canada/Eastern	EST/EDT	-5 hours	Canadian
Canada	Canada/Central	CST/CDT	-6 hours	Canadian
Canada	Canada/Mountain	MST/MDT	-7 hours	Canadian
Canada	Canada/Pacific	PST/PDT	-8 hours	Canadian
Canada	Canada/Yukon	YST/YDT	-9 hours	Canadian
Canada	Canada/ Saskatchewan	CST	-6 hours	-none-
Israel	Israel	IST/DST	+2 hours	Israeli
Australia	Australia/ Tasmania	EST	10 hours	Australian
Australia	Australia/ Queensland	EST	10 hours	-none-
Australia	Australia/North	CST	9:30 hours	-none-
Australia	Australia/West	WST	8 hours	-none-
Australia	Australia/South	CST	9:30 hours	Australian

Table 10-1 Loadable Timezone Rules (Continued)

Country Name	Rule Name	Timezone Name	GMT Offset	DST Rules
Australia	Australia/Victoria	CST	10 hours	Australian
Australia	Australia/NSW	CST	10 hours	Australian
Australia	Australia/ Yarcowinna	CST	9:30 hours	Australian
Australia	Australia/LHI	CST	10:30 hours	Australian
Europe	Britain[a]	GMT/BST	0 hours	GB-Eire
Europe	Europe/Western[a]	WET/WET-DST	0 hours	W-Eur
Europe	Europe/Middle[a]	MET/MET-DST	1 hour	M-Eur
Europe	Europe/Central[a]	CET/CET-DST	1 hour	M-Eur
Europe	Europe/Eastern[a]	EET/EET-DST	2 hours	E-Eur
	Iceland	GMT	1 hour	-none-
	Poland	MET	1 hour	W-Eur
	Turkey	EET/EET/DST	3 hours	Turkey
Japan	Japan[a]	JST	+9 hours	-none-
Singapore	Singapore[a]	SST	+8 hours	-none-
NewZealand	NewZealand[a]	NZST/NZDT	+12 hours	New Zealand

a.This timezone is also compiled-in.

TIMEDC Command Reference

The TIMEDC command returns information about the `timed` daemon, such as the time difference between the local clock and that of a specified host or hosts, the location of the primary server, and logged TIMED messages in the TCPWARE:TIMED.LOG file. You can issue this command from any machine running TCPware's TIMED. The following page provides a TIMEDC command reference.

TIMEDC

Returns information about the `timed` daemon on the local host. You can issue this command from any machine running TCPware's TIMED.

Execute the command on the DCL level.

Format

TIMEDC *[option]*

The options for the TIMEDC command follow. If you omit the option, you are prompted for it at the `timedc>` prompt.

Options

HELP
?

Provides online help.

CLOCKDIFF *host [host ...]*

Computes the difference (in milliseconds) between the clocks of the local and specified host or hosts or IP address or addresses (with multiple entries separated by spaces). Synchronization can occur to within 20 milliseconds.

DATE *[yyyy [mm[dd[hh]]]]mm[ss]*

Sets the networking time to the specified date and time, in the order *year, month*, *date, hour*, *minutes,* and *seconds*. Entry requires at least *minutes* (the remaining values are optional).

MSITE

Finds the location of the primary time server.

TRACE {ON | OFF}

Enables or disables tracing of incoming messages to the `timed` daemon. The TCPWARE:TIMED.LOG file holds the messages.

QUIT

Quits the TIMEDC command.

Examples

1 Gives the time difference in milliseconds between HOMER and local host BART.

```
$ TIMEDC CLOCKDIFF HOMER
time on HOMER.PROCESS.COM is 124 ms. ahead of time on
bart.process.com
```

2 Locates the primary timed daemon, which in this case is MARGE.

```
$ TIMEDC MSITE
master timedaemon at homer.process.com is marge.process.com
```

3 PENNY is not a TIMED host or its `timed` process is not running.

```
$ TIMEDC
timedc> CLOCKDIFF PENNY
penny.process.com is down
```

PART V Managing Applications

Chapter 11

Managing FTP-OpenVMS

Introduction

This chapter describes FTP-OpenVMS Client and FTP-OpenVMS Server management. Topics include:

- Client considerations
- Server security considerations
- Including messages on client login
- ANONYMOUS support
- Server logicals
- Server FTP protocol implementation

Client Considerations

Client considerations include creating a startup command file and determining the status on an exit condition.

Startup Command File

You can create a system-wide startup file that is executed each time the local client starts an FTP session. To create a startup command file, you need to:

1	Create a file containing the FTP commands you want performed at the beginning of each FTP session . For example:
	`$ CREATE SYS$COMMON:[SYSMGR]FTP_STARTUP.COM`
	`OPEN IRIS SMITH "Sandy"`
	`SHOW STATUS`
	`Ctrl/Z`
	`$ SET PROTECTION=WORLD:RE SYS$COMMON:[SYSMGR]FTP_STARTUP.COM`
	The SET PROTECTION command ensures that the client user can read and execute the FTP_STARTUP.COM file.

2	Define the FTP_STARTUP logical to point to the FTP_STARTUP.COM file. For example: `$ DEFINE/SYSTEM/EXEC FTP_STARTUP SYS$MANAGER:FTP_STARTUP.COM` Client users can override this startup file by creating their own. Including the command DEFINE/PROCESS FTP_STARTUP in a user's LOGIN.COM file overrides any DEFINE/SYSTEM/EXEC command in the SYS$MANAGER:SYSTARTUP_V5.COM file.

See the *User's Guide*, Chapter 3, *FTP: Transferring Files*, *Startup Command File* about setting up a client-specific FTP_STARTUP.COM file.

SET DEBUG /CLASS=REPLIES (VERBOSE mode) is enabled by default in TCPware's FTP-OpenVMS Client so that you do not need to explicitly add the command to the file. However, any VERBOSE command may toggle it to OFF.

See the *User's Guide*, Chapter 3, *FTP: Transferring Files*, about setting VERBOSE mode. See your OpenVMS documentation about the SET PROTECTION command.

Status on Exiting FTP Status

To exit FTP, use the **EXIT** command or type `Ctrl/Z`

FTP exits with the last error status, if any. DCL command procedures can use the $STATUS and $SEVERITY symbols to test for success or failure of the FTP commands issued. A success status indicates that all commands succeeded. A warning, error, or severe status indicates that one or more commands failed to execute.

When possible, the status code is a System Service (defined in $SSDEF), RMS (defined in $RMSDEF), or shared (defined in $SHRDEF) status. In some cases, status codes are TCPWARE_ private codes with a facility number of 1577.

Server Security

The FTP server provides security through login procedures, the use of log files, and the automatic termination of idle control connections.

Login Procedures

The FTP server uses the same login procedures as DECnet network connections and does not support OpenVMS accounts with two passwords.

Directory Access Restrictions

The FTP server lets you define three logicals for access restrictions to specific directory trees. These include the TCPWARE_ FTP_ROOT logical for system-wide access restrictions, the TCPWARE_FTP_ANONYMOUS_ROOT logical for ANONYMOUS user access restrictions, and the TCPWARE_FTP_*username*_ROOT logical for specific username access restrictions. See *Server Logicals* and *ANONYMOUS Support*.

Log File

The FTP server creates a log file in your default directory each time a client user successfully logs in. This FTPSERVER_DTP.LOG file contains information about files transferred during the FTP session. If client users have problems logging in and are sure they specified a proper user name and password, you can check the SYSLOGIN and user account login command procedures for commands that could have caused the login to fail.

Examining the FTPSERVER_DTP.LOG file might help isolate the problem. You may need to execute some operations only if the process mode is interactive. (Use the F$MODE() lexical function to determine the mode and then skip around the offending commands if not an interactive login.) The FTP server runs in network mode.

If you suspect break-in attempts, you can also define the TCPWARE_FTP_LOGFILE system logical to specify the name of a log file. See *TCPWARE_FTP_LOGFILE*.

Idle Control Connection Timeout

If the control connection (other than during a data transfer) is idle for more than 10 minutes, the FTP server aborts the connection, unless you change the idle timeout value using the TCPWARE_FTP_IDLE_TIMEOUT logical. See *TCPWARE_FTP_IDLE_TIMEOUT*.

Special Messages

You can include special informational text messages in a specified file in directories so that the message appears when an FTP client user logs in or changes to that directory. The TCPWARE_FTP_MESSAGE_FILE logical determines the filename to check in each directory. This feature is particularly helpful for ANONYMOUS FTP client users to get informational messages when changing directories (see the next section for a description of ANONYMOUS support).

For example, the FTP_CONTROL.COM file that you activate on startup includes the following line commented-out:

```
!$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_MESSAGE_FILE ".MESSAGE"
```

You can activate this logical with the default `.MESSAGE` file definition, or change it to WELCOME.TXT, for example. What the file should contain depends on the context. If the file is in a login directory, it should contain a general message about the system, such as `Welcome to OpenVMS VAX V6.2 (IRIS)`. If the file is in another directory to which the user can move, it should have specifics about the directory, such as `Welcome to the SUPPORT directory. It contains TECHNICAL SUPPORT information`.

The FTP client user must set VERBOSE mode to be able to see the messages. VERBOSE (REPLIES) mode is set by default in TCPware's FTP-OpenVMS Client.

The FTP command line client interprets the exclamation point (!) as the start of a comment. To send an exclamation point to the server it must be enclosed in quotes ("). The quote character (") must be doubled up in the string if it is to be sent to the server. For example:

```
FTP> “! send this string to the server.”
```

sends the whole line, including the exclamation point and the period.

The FTP command line client prompts with the name of the opened node when the TCPWARE_FTP_PROMPT_NODENAME logical is defined.

ANONYMOUS Support

The FTP server provides special support for ANONYMOUS accounts.

To set up an ANONYMOUS account, issue commands using the OpenVMS AUTHORIZE utility:

```
$ AUTHORIZE

UAF> ADD ANONYMOUS/PASSWORD=GUEST/UIC=[uic] -
_UAF> [/other qualifiers] /NOPWDEXP/NOPWDLIFE
UAF> MODIFY ANONYMOUS/NOLOCAL/NOBATCH/NOREMOTE/NODIALUP
UAF> MODIFY ANONYMOUS/PRIV=NONETMBX/DEFPRIV=NONETMBX
```

The /NOPWDEXP and /PWDLIFE=NONE qualifiers ensure that the password remains active indefinitely. The /LOCAL, /NOBATCH, /NOREMOTE, and /NODIALUP qualifiers prevent access to the account from those sources. Removing the NETMBX privilege prevents DECnet access.

See your OpenVMS documentation for details on the AUTHORIZE utility.

FTP users have automatic read access through ANONYMOUS accounts. You can also assign write, rename, or delete access by defining the TCPWARE_FTP_ANONYMOUS_RIGHTS logical.

See *TCPWARE_FTP_ANONYMOUS_RIGHTS*.

Be aware of the following:

- You must create the ANONYMOUS account to use the password GUEST. If the account has any other password, users cannot log in. If the account has another password, use the

MODIFY ANONYMOUS /PASSWORD=GUEST/NOPWDEXP command in the AUTHORIZE utility to change it.

- It is recommended that the ANONYMOUS account use the rooted logical (such as ANONYMOUS_ROOT) to point to the top level directory and set the default to ANONYMOUS_ROOT:[000000]. In this way, when a client changes the directory to a slash (`/`) (or uses a slash in front of a directory specification, as is the case with some WWW browsers), SYS$DISK:[000000] maps to this root directory. (Note that this mapping is independent of the access restrictions applied by the TCPWARE_FTP_*_ROOT logicals.)
- If you define the TCPWARE_FTP_ANONYMOUS_ROOT logical (or the TCPWARE_FTP_ROOT logical on a system-wide level), the system restricts ANONYMOUS user access to files in the root directory and subdirectories only. When the ANONYMOUS user logs in, the root directory becomes the default.
- In response to a username, the system sends a reply message of `331 Send ident as Password` instead of the usual `331 Password required`.
- If you define the TCPWARE_FTP_ANONYMOUS_230_REPLY logical, the system uses it to generate the reply message when the ANONYMOUS user logs in. If you do not define this logical, the system uses the default 230 reply instead.
- Using a hyphen (`-`) as the first character of the password causes the system to turn off the message generated by the logicals. The system sends the default 230 reply instead.
- If you define the TCPWARE_FTP_LOGFILE logical, the server writes a slightly different record to the log file:

  `date/time ANONYMOUS FTP login successful (password) from ia, port`

  Each command is logged with the following format:

  `date/time ANONYMOUS FTP user (password) at ia, job 120 reply to user command job 120 PWD`

  The response to each command is logged with the following format:

  date/*time* ANONYMOUS FTP user (*password*) from *(internet address)*, job (*pid*), (*reply string*) or

  date/*time* user name from *(internet address)*, job (*pid*), (*reply string*)

- SITE SPAWN is not a valid command with ANONYMOUS FTP.

Server Logicals

The FTP server supports the following special system-definable logicals:

- TCPWARE_FTP_220_REPLY
- TCPWARE_FTP_221_REPLY
- TCPWARE_FTP_230_REPLY
- TCPWARE_FTP_421_REPLY
- TCPWARE_FTP_ALL_VERSIONS
- TCPWARE_FTP_ALLOWCAPTIVE
- TCPWARE_FTP_ANONYMOUS_230_REPLY

- TCPWARE_FTP_ANONYMOUS_RIGHTS
- TCPWARE_FTP_ANONYMOUS_ROOT
- TCPWARE_FTP_DISALLOW_UNIX_STYLE
- TCPWARE_FTP_EXTENSION_QUANTITY
- TCPWARE_FTP_FAST_TIMEOUT
- TCPWARE_FTP_IDLE_TIMEOUT
- TCPWARE_FTP_KEEP_DIR_EXT
- TCPWARE_FTP_LOGFILE
- TCPWARE_FTP_LOG_ALL_USERS
- TCPWARE_FTP_MAXREC
- TCPWARE_FTP_MAX_SERVERS
- TCPWARE_FTP_MESSAGE_FILE
- TCPWARE_FTP_ONLY_BREAK_ON_CRLF
- TCPWARE_FTP_RECEIVE_THRESHOLD
- TCPWARE_FTP_RECODE_NONVMS_FILE_NAMES
- TCPWARE_FTP_ROOT
- TCPWARE_FTP_*username*_ROOT
- TCPWARE_FTP_SEMANTICS_FIXED_IGNORE_CC
- TCPWARE_FTP_SERVER_DATA_PORT_RANGE
- TCPWARE_FTP_SERVER_LOG_LIMIT
- TCPWARE_FTP_SERVER_RELAXED_PORT_COMMAND
- TCPWARE_FTP_STRIP_VERSION
- TCPWARE_FTP_SYST_BANNER
- TCPWARE_FTP_WINDOW
- TCPWARE_FTP_UNIX_STYLE_BY_DEFAULT
- TCPWARE_FTP_UNIX_STYLE_CASE_INSENSITIVE

Note! With the root logicals (TCPWARE_FTP_ROOT, TCPWARE_FTP_ANONYMOUS_ROOT, and TCPWARE_FTP_*username*_ROOT), any logical you refer to in the equivalence name (such as a disk name) must also be an executive mode, system table logical. With all of these logicals, if the user account cannot access the directory, FTP operations will fail with the error `%RMS-E-PRV`.

TCPWARE_FTP_220_REPLY

The TCPWARE_FTP_220_REPLY logical defines a message displayed when a user connects to the server and can log in. This message replaces the default message.

You can define lines of the message text, one comma-separated equivalence string for each line. You can also specify a file that contains the message text by defining an equivalence string starting with the at-sign (@) and followed by the complete file specification. For example, you can define the welcome text equivalence string as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_220_REPLY -
```

```
_$ "**AUTHORIZED USE ONLY **",-
_$ "bart.nene.com (192.168.34.56)", -
_$ "FTP-OpenVMS FTPD V5.4 (c) 1999 Process Software Corporation"
```

Alternately, you can include the last three equivalence strings in an FTP_WELCOME.TXT file and define the logical as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_220_REPLY -
_$ "@SYS$MANAGER:FTP_WELCOME.TXT"
```

In either case, when a user connects to a host, the message appears as follows:

```
220-** AUTHORIZED USE ONLY **
220-bart.nene.com (192.168.34.56)
220 FTP-OpenVMS FTPD V5.4 (c) 1999 Process Software Corporation
_Username []:
```

TCPWARE_FTP_221_REPLY

The TCPWARE_FTP_221_REPLY logical defines a message to appear when a user ends the FTP session. If you do not define this logical, TCPware uses the default message instead. As with TCPWARE_FTP_220_REPLY, you can define a text string or file. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_221_REPLY -
_$ "Connection to FTP server has been closed"
```

Now, when the user closes the FTP connection, the following message appears:

```
221 Connection to FTP server has been closed
```

TCPWARE_FTP_230_REPLY

The TCPWARE_FTP_230_REPLY logical defines a message to appear when a user successfully logs in. If you do not define this logical, TCPware uses the default message instead. As with TCPWARE_FTP_221_REPLY, you can define a text string or file. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_230_REPLY-
_$ "Login successful"
```

Now, when the user logs in using FTP, the following message appears:

```
230 Login successful
```

TCPWARE_FTP_421_REPLY

The TCPWARE_FTP_421_REPLY logical defines a message sent when a user connects to the server but should not log in. After sending the message, the connection closes. For example, you can define this logical to prevent FTP access for a short time period. Be sure to deassign the logical after this period to allow FTP access again. As with TCPWARE_FTP_230_REPLY, you can define a text string or file. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_421_REPLY-
_$ "System maintenance in progress until 17:30"
```

Now, when the user connects to the host through FTP, the following message appears and then the connection closes:

```
421 System maintenance in progress until 17:30
```

Note! The TCPWARE_FTP_421_REPLY logical has precedence over the TCPWARE_FTP_220_REPLY logical.

TCPWARE_FTP_ALL_VERSIONS

The logical name TCPWARE_FTP_ALL_VERSIONS requests the NLST and LIST commands to display all versions of the specified files. If TCPWARE_FTP_ALL_VERSIONS is defined, the logical name TCPWARE_FTP_STRIP_VERSION has no effect.

Note! TCPWARE_FTP_ALL_VERSIONS is ignored if the FTP Server is in UNIX emulation mode.

TCPWARE_FTP_ALLOWCAPTIVE

By default, the FTP server does not allow file transfers for CAPTIVE accounts. However, by defining the TCPWARE_FTP_ALLOWCAPTIVE logical, you can allow CAPTIVE accounts to use all FTP commands except SITE SPAWN. Define the logical as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_ALLOWCAPTIVE " "
```

You must also modify the CAPTIVE account procedure to allow the FTP server to start the data transfer process. The procedure can check if the logical "TT" is equal to "TCPWARE:FTPSERVER_DTP.COM" and exit out of the login procedure, as follows:

```
$! Check if this is the TCPware FTP data transfer process:
$ IF F$LOGICAL("TT") .EQS. "TCPWARE:FTPSERVER_DTP.COM" THEN EXIT
$! Refuse other network connections (such as DECnet):
$ IF F$MODE() .EQS. "NETWORK" THEN LOGOUT
$! (or allow by using "...THEN EXIT" above)
$! Remainder of CAPTIVE procedure follows:
$....
```

TCPWARE_FTP_ANONYMOUS_230_REPLY

The TCPWARE_FTP_ANONYMOUS_230_REPLY logical defines a message to appear when an ANONYMOUS user successfully logs in. If you do not define this logical, TCPware uses the default message instead. As with TCPWARE_FTP_230_REPLY, you can define a text string or file. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_ANONYMOUS_230_REPLY-
_$ "ANONYMOUS login successful"
```

Now, when a user logs in using the ANONYMOUS account, the following message appears:

```
230 ANONYMOUS login successful
```

TCPWARE_FTP_ANONYMOUS_RIGHTS

The TCPWARE_FTP_ANONYMOUS_RIGHTS logical lets you define write, rename, and delete access rights for the ANONYMOUS FTP user in addition to read access. For example:

```
$ DEFINE/SYS/EXEC/NOLOG TCPWARE_FTP_ANONYMOUS_RIGHTS-
_$ "WRITE,RENAME,DELETE"
```

WRITE	Lets you PUT, COPY, SEND, and MPUT files into the ANONYMOUS FTP area. It also allows execution of the CREATE/DIRECTORY command.
RENAME	Lets you rename and append files in the ANONYMOUS FTP area.
DELETE	Lets you delete files and directories from the ANONYMOUS FTP area.

The definition of these rights does not override the actual file protections. If a directory does not allow write access, users cannot write to the directory even though the TCPWARE_FTP_ANONYMOUS_RIGHTS logical grants them write access. Likewise, if a file does not allow delete access, users cannot delete it even if the TCPWARE_FTP_ANONYMOUS_RIGHTS logical grants them delete access.

TCPWARE_FTP_ANONYMOUS_ROOT

The TCPWARE_FTP_ANONYMOUS_ROOT (system level, executive mode) logical defines access restrictions for users logged in as ANONYMOUS. For example, you can set access restrictions for users logged in as ANONYMOUS to allow access to just the ANONYMOUS$USER directory and its subdirectories, as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_ANONYMOUS_ROOT ANONYMOUS$USER:
```

If you do not set this logical, the FTP server defaults to the setting in the TCPWARE_FTP_ROOT logical (described above), if it exists.

TCPWARE_FTP_DISALLOW_UNIX_STYLE

This controls whether UNIX style filename parsing is done. If this logical is not defined and a / is found in the filename, then it is assumed to be a UNIX style filename. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_DISALLOW_UNIX_STYLE ?
```

TCPWARE_FTP_EXTENSION_QUANTITY

Defines the default allocation /extention quantity for new files and appends. See FAB$W_DEQ in the OpenVMS Record Management Services Manual for an explanation of the effect of this. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_EXTENSION_QUANTITY ?
```

TCPWARE_FTP_IDLE_TIMEOUT

If you want to change the timeout for FTP connection attempts to something other than the default of 10 minutes, use the TCPWARE_FTP_IDLE_TIMEOUT system logical. The FTP server checks the timeout when you enter and complete a command. Therefore, you can set this logical any time, and it effectively changes the idle timeout for open, non-idling connections as well as for any future ones. Make sure to use delta time for the time syntax. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_IDLE_TIMEOUT "0 00:20:00"
```

This example changes the idle timeout to 20 minutes. If omitted, the default is 10 minutes. If you set the value to 0, idle timeout is disabled.

TCPWARE_FTP_KEEP_DIR_EXT

Sometimes the FTP server strips the .DIR extension from the file name of a directory when the NLST function is requested. The FTP server now looks for the logical TCPWARE_FTPD_KEEP_DIR_EXT and, if defined, does not remove the .DIR extension. To use this feature, define the system/exe mode logical:

```
$ DEFINE/SYSTEM/EXE TCPWARE_FTPD_KEEP_DIR_EXT TRUE
```

To return to the default behavior, remove this logical.

TCPWARE_FTP_LOGFILE

The TCPWARE_FTP_LOGFILE (system level, executive mode) logical can be defined to specify the name of a log file. This is good if you suspect break-ins to the FTP server. For example:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_LOGFILE-
_$ SYS$COMMON:[SYSMGR]FTPLOGIN.LOG
```

If this logical exists, the FTP server writes a record to the specified file each time a user attempts to log in. Each record includes the date and time, the remote host’s internet address, and whether the login succeeded.

This logical specifies the name of the file to which ALL commands and responses to ANONYMOUS FTP services are logged. If TCPWARE_FTP_LOG_ALL_USERS is also defined, then commands and responses for all users are logged.

TCPWARE_FTP_MAXREC

The FTP client and the FTP server normally check the record size of an ASCII transfer and disallow more than 8192 byte records (as a sanity check). However, you can define the TCPWARE_FTP_MAXREC logical to override the default of 8192. The definition of the TCPWARE_FTP_MAXREC logical is commented out but defined in the FTP_CONTROL.COM file as follows:

```
$ !DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_MAXREC 8192
```

TCPWARE_FTP_MAX_SERVERS

The logical name TCPWARE_FTP_MAX_SERVERS allows the maximum number of servers to be set. The default is 10000.

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_MAX_SERVERS "1500"
```

TCPWARE_FTP_MESSAGE_FILE

The TCPWARE_FTP_MESSAGE_FILE logical defines the message file the FTP user sees when connecting to the server or moving between directories. The definition of the TCPWARE_FTP_MESSAGE_FILE logical is commented out but defined in the FTP_CONTROL.COM file as follows:

```
$ !DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_MESSAGE_FILE ".MESSAGE"
```

See *Special Messages*.

TCPWARE_FTP_ONLY_BREAK_ON_CRLF

If the TCPWARE_FTP_ONLY_BREAK_ON_CRLF logical is set and an ASCII file is transferred, a new line is created in the file upon receipt of a carriage return/line feed sequence.

If this logical is not set and an ASCII file is transferred, a new line is created upon receipt of either a carriage return/line feed sequence or a line feed.

TCPWARE_FTP_RECEIVE_THRESHOLD

The TCPWARE_FTP_RECEIVE_THRESHOLD logical specifies the amount of buffer space that can be used to buffer transmitted data on the data socket. The default value if 6144. If this logical is defined and it begins with a /, then it specifies the fraction of the window size; if only a fraction is specified, then it indicates the number of bytes to be used. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE TCPWARE_FTP_RECEIVE_THRESHOLD ?
```

TCPWARE_FTP_ROOT

The TCPWARE_FTP_ROOT (system level, executive mode) logical defines the system-wide default directory access restrictions for client users. For example, you can restrict all users logged in via FTP to the COMMON$USER directory and its subdirectories, as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_ROOT COMMON$USER:
```

The FTP server defaults to this logical if the TCPWARE_FTP_ANONYMOUS_ROOT or TCPWARE_FTP_*username*_ROOT logicals (described in the next section) are not set.

TCPWARE_FTP_*username*_ROOT

The TCPWARE_FTP_*username*_ROOT (system level, executive mode) logical defines access restrictions for an FTP client logging in as *username*. For example, you can restrict user CLARK to the COMMON$USER:[CLARK] directory and its subdirectories, as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_CLARK_ROOT COMMON$USER:[CLARK]
```

Because the FTP server restricts access by default to the directory setting in the TCPWARE_FTP_ROOT logical (described earlier), if it exists, you may want to use the special wildcard (*****) setting with the TCPWARE_FTP_*username*_ROOT logical to bypass the default for *username*. For example, to restrict the bulk of users to DISK$SYS_LOGIN, restrict users KATE and PAUL to ENG$DISK, but allow SYSTEM full access to locations covered by its account, define the following logicals:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_ROOT DISK$SYS_LOGIN ! default
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_KATE_ROOT ENG$DISK  ! limits KATE
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_PAUL_ROOT ENG$DISK  ! limits PAUL
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_SYSTEM_ROOT *       ! full SYSTEM
```

ANONYMOUS user access restrictions are described under *TCPWARE_FTP_ANONYMOUS_ROOT.*

TCPWARE_FTP_SEMANTICS_FIXED_IGNORE_CC

If the TCPWARE_FTP_SEMANTICS_FIXED_IGNORE_CC logical is defined to TRUE, then GET operations of fixed lengths record files will not have a <CR>(carriage return)<LF>(line feed) added to the end of each record. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE TCPWARE_FTP_SEMANTICS_FIXED_IGNORE_CC ?
```

TCPWARE_FTP_SERVER_DATA_PORT_RANGE

This specifies the upper and lower port boundaries that are to be used in passive data connections. The string should contain two numbers separated by a space. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE TCPWARE_FTP_SERVER_DATA_PORT_RANGE ?
```

TCPWARE_FTP_SERVER_LOG_LIMIT

By setting the logical name TCPWARE_FTP_SERVER_LOG_LIMIT in the LOGIN.COM file, you can specify that log files be retained. Set the logical name to a dash (-) to retain all log files, or specify a number in the range of 1 to 32000.

Directory size restrictions limit the number of potential files that can actually be created. If you do not specify a number or value, one log file is created or overwritten for each FTP session. Use the DCL PURGE command to delete unneeded log files. The following example specifies that 42 log files be retained:

```
$ DEFINE TCPWARE_FTP_SERVER_LOG_LIMIT 42
```

TCPWARE_FTP_SERVER_RELAXED_PORT_COMMAND

The server normally compares the IP network address value specified in the PORT command with the IP network address of the IP address that it is receiving commands from. If these are not in agreement, the PORT command is not accepted. Some multi-homed clients, and clients that can do third party transfers send values that do not match. Defining this logical allows the PORT command to be accepted for these clients by disabling this check. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE TCPWARE_FTP_SERVER_RELAXED_PORT_COMMAND ?
```

TCPWARE_FTP_STRIP_VERSION

The logical name TCPWARE_FTP_STRIP_VERSION causes VMS mode output to have no versions. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_STRIP_VERSION ?
```

TCPWARE_FTP_SYST_BANNER

If the FTP Server is in UNIX mode, the SYST command displays the banner “UNIX TCPware Unix Emulation.” If the FTP Server is in VMS mode, the SYST command displays the equivalence string associated with the TCPWARE_FTP_SYST_BANNER logical name (if defined). Otherwise, the SYST command displays “VMS TCPware V*x.y(rev)*,” where:

– V*x.y* is the TCPware version number.
– *(rev)* is the revision number of the FTP Server.

The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_SYST_BANNER ?
```

Note! The logical name TCPWARE_FTP_SYST_BANNER is ignored if the FTP Server is already in UNIX mode.

TCPWARE_FTP_UNIX_STYLE_BY_DEFAULT

If you define the logical name TCPWARE_FTP_UNIX_STYLE_BY_DEFAULT, the FTP Server starts in UNIX emulation mode.

The control of version number displays has been reworked in response to LIST and NLST commands. The default is VMS-mode output. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_UNIX_STYLE_BY_DEFAULT ?
```

When sending the command from a non-OpenVMS client, a space is required between the file specification and the qualifier. For example:

```
$ GET filename /LOG
```

Previous command syntax: ftp>**put xx x.x/image=2048**
New command syntax: ftp>**put x.x “x.x/image=2048”**

You can disable this feature so that the FTP server can accept an OpenVMS transfer mode qualifier without including the space between the file specification and the qualifier. To disable this requirement, define the following logical:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE TCPWARE_FTPD_NOUNIX_SYNTAX “TRUE”
```

TCPWARE_FTP_UNIX_STYLE_CASE_INSENSITIVE

The logical name TCPWARE_FTP_UNIX_STYLE_CASE_INSENSITIVE allows UNIX style filename handling to be case insensitive. The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_UNIX_STYLE_CASE_INSENSITIVE ?
```

TCPWARE_FTP_WINDOW

The FTP client and the FTP server set the TCP window size of the data connection to either:

- The value of the TCPWARE_FTP_WINDOW logical if you define it (the minimum value is 512 bytes, the maximum value is 1,048,576 bytes).
- The larger of 32,768 bytes and the default TCP window size.

The ? in the logical represents where defined values go. Defined value can be either alpha or numeric.

```
$ DEFINE/SYSTEM/NOLOG/EXEC TCPWARE_FTP_WINDOW ?
```

Implementation

This section describes the FTP server implementation of the File Transfer Protocol (FTP) as defined in the RFC 959. The material in this section requires a thorough understanding of the protocols used.

The FTP server is now more “UNIX friendly” and accommodates pathname specifications in some Web browsers; the forward slash (/) at the beginning of directory structures is now recognized.

The FTP server implements the following FTP service commands defined in the FTP protocol:

ACCT *arguments*	TCPware ignores this command but acknowledges it with successful completion.
ALLO *arguments*	If specified before a STOR operation, rounds up the size (in bytes) specified with ALLO to the number of blocks and uses it as the initial allocation. If the size specified with ALLO is negative, the allocation is contiguous.

APPE *filespec*	Appends the data received from the requesting host to the specified file (if the file does not exist, TCPware creates it). (See also *RETRIEVE, STORE, and APPEND Command Qualifiers*.)
CDUP	Sets the default working directory to the parent directory for the current directory. XCUP is a synonym.
CWD *directory*	Specifies the new default working directory. XCWD is a synonym.
DELE *filespec*	Deletes the file (or files) specified.
HELP *[topic]*	Requests help information.
LIST *filespec*	Returns a directory listing.
MKD *filespec*	Creates the specified directory. **XMKD** is a synonym.
MODE *arguments*	Specifies the transfer mode. The valid arguments are **S** (STREAM) and **C** (COMPRESSED).
NLST *filespec*	Returns a list of file names without a .DIR extension if the client is not an OpenVMS machine. Retrieving a directory file (*.DIR) if the client is not an OpenVMS machine results in an error message.
NOOP	TCPware ignores this command but acknowledges it with successful completion.
PASS *password*	Logs the user into the host. If the first character of the password is a hyphen (**-**), the default successful login (230) message appears. The optional messages defined by the TCPWARE_FTP_230_REPLY or TCPWARE_FTP_ANONYMOUS_230_REPLY logicals do not appear. This supports clients that cannot receive the multi-line replies these logicals can generate.
PASV	Requests the server data transfer process to be passive. This means to "listen" on a non-default data port and wait for a connection instead of initiating one upon receiving a transfer command. The server responds with the host name and port number on which it is listening.
PORT *arguments*	Specifies the data port number used for data transfers. The FTP server reports a 501 Bad parameter value error if a port less than 1024 is specified. If you want to use a privileged port for the destination of data transfer, define the following logical to disable this feature: `$ DEFINE/SYSTEM TCPWARE_FTPD_ALLOW_PRIV_PORT “TRUE”`
PWD	Returns the current working directory. **XPWD** is a synonym.
QUIT	Closes the connection.

REIN	Logs out the user and resets the file transfer parameters to the initial values.
RETR *filespec*	Reads the file and transmits it to the requesting host. (See also *RETRIEVE, STORE, and APPEND Command Qualifiers*.)
RMD *filespec*	Deletes the specified directory if the directory is empty. **XRMD** is a synonym. An error reply is sent if the directory is not empty.
RNFR *filespec*	Specifies the file to be renamed.
RNTO *filespec*	Specifies the new name of the file designated in the RNFR command.
SITE *arguments*	Used for site-specific requirements or capabilities. The following SITE commands are supported: **SITE HELP**--Returns a list of supported SITE commands. **SITE PRIV** *[privileges]*--Turns process privileges on or off. The arguments are ALL, NONE, or a privilege name. With no arguments, SITE PRIV displays the current process privileges. **SITE RMS RECSIZE** *[value]*--Controls the record size used when writing binary files; any valid RMS record size value is permitted. With no arguments, displays the current value. Applies only when STREAM is OFF. The default is 512. **SITE SHOW TIME**--Returns the current date and time-of-day for the OpenVMS system in the reply message. **SITE SPAWN** *command-line*--A subprocess executes the specified command line. Use this command for submitting batch jobs and printing files. The status returned for the SITE command depends on the status returned by the utility or command executed (see the VMS documentation regarding the DCL $STATUS symbol). **SITE +VMS+**--Receiving this command from a Compaq TCP/IP Services for OpenVMS (UCX) client sets the file transfer mode to VMS_PLUS. **WINDOW_SIZE**--Displays or sets the TCP window size.
STOR *filespec*	Writes the file from data received from the requesting host. The STORE command supports the /ASCII, /BINARY, /BLOCK, /CONTIGUOUS=*blocks*, /FORTRAN, /IMAGE[=*n*], /RECORD, /VARIABLE, and /VMS qualifiers. (See also *RETRIEVE, STORE, and APPEND Command Qualifiers*.)

STOU *filespec*	Writes the data received from the requesting host to a unique filename. If you specify a *filespec*, TCPware uses it as the seed for the unique filename; otherwise, the server creates a unique filename. The STOU command uses a data connection.

Note! The STOU *filespec* pathname can contain the /ASCII, /BINARY, /BLOCK, /FORTRAN, or /IMAGE qualifier to specify the transfer mode. A qualifier can cause unpredictable results depending on the current TYPE and STRU settings. The pathname can also contain /CONTIGUOUS=*blocks*, in which case TCPware creates the file with an initial contiguous allocation of the specified number of blocks.

STRU *arguments*	Specifies the file structure. The valid arguments are **F** (for *file*, or no record structure), **R** (for *record* structure), and **O VMS** (or **VMS**, for VMS file structure). For VMS file structure, the data sent over the connection consists of a small header containing RMS file information, followed by raw data from the file, block by block.
SYST	Returns the name of the operating system running on the server.
TYPE *arguments*	Specifies the file type. The valid arguments are **A** (ASCII), **I** (image), **L 8** (image), **A N** (ASCII non-print), **A C** (FORTRAN carriage control), and **A T** (Telnet format effectors). Use I (image) for both formatted binary and image format transfers. Formatted binary data includes the necessary record headers and checksums.
USER *name*	Logs the user into the host.

All other commands result in error 500 or 502 (command not implemented). This implementation accepts and may issue all response codes.

RETRIEVE, STORE, and APPEND Command Qualifiers

The FTP server accepts the following qualifiers with client commands that send RETR, STOR, and APPE commands:

Note! When using the /ASCII, /BINARY, /BLOCK, /FORTRAN, or /IMAGE qualifier with commands that send RETR or STOR commands, make sure to separate the file specification and the qualifier with a space character. Otherwise the qualifier can be considered part of a UNIX file specification. For example, use the following on the client:

```
ftp> put sample.txt "sample.txt /block"
```

/ASCII	TCPware reads or writes the file as an ASCII file.

/BINARY	TCPware reads or writes the file as a formatted binary file. Use this qualifier when transferring variable length binary files that do not have a file extension of .OBJ, .STB, .BIN, or .LDA.
/BLOCK	TCPware reads or writes the file using block-I/O mode. Use this qualifier when transferring STREAM_LF, STREAM_CR, STREAM or UNDEFINED files.
/CONTIGUOUS	(Applies to STOR only) the local output file should have an initial contiguous allocation of the specified number of blocks. If the output file is smaller, the FTP server truncates it. If the output file is larger, the additional allocations are noncontiguous.
/FORTRAN	TCPware reads or writes the file as a FORTRAN carriage control file.
/IMAGE*[=n]*	TCPware reads or writes the file as an image file. If you specify a record length, it only applies to output files.
/VARIABLE	TCPware writes an image format file as a variable length record format file. Ignored for all other transfer formats.

The FTP server also supports the STRU O VMS (or STRU VMS) format that allows OpenVMS systems to exchange any RMS file, including RMS indexed files.

Note! Some combinations of these qualifiers and the TYPE and STRU commands may produce unpredictable results. Use these qualifiers carefully.

Troubleshooting

Q:How can I apply Access Control Lists (ACLs) to my FTP-OpenVMS executables so that only I have access?

A:Assume you want to set up your username as FTP_USER and give yourself (and no one else) read and execute privileges to the FTP-OpenVMS executables:

```
$ SET DEFAULT$SYSTEM
$ MCR AUTHORIZE
UAF> ADD/ID FTP_USER
UAF> GRANT/ID FTP_USER yourname
Then, for the FTP-OpenVMS Client:
$ SET DEFAULT TCPWARE
$ EDIT/ACL FTP.EXE

(IDENTIFIER=FTP_USER,ACCESS=READ+EXECUTE)
(IDENTIFIER=*,ACCESS=NONE)
Ctrl/Z
```

```
For the FTP-OpenVMS Server:
$ SET DEFAULT TCPWARE
$ EDIT/ACL FTP_DTP.EXE

(IDENTIFIER=FTP_USER,ACCESS=READ+EXECUTE)
(IDENTIFIER=*,ACCESS=NONE)
Ctrl/Z
```

Chapter 12

Managing NFS-OpenVMS Client

Introduction

This chapter describes how to manage the NFS-OpenVMS Client. Topics include client management concepts and how to mount remote filesystems.

Client Concepts

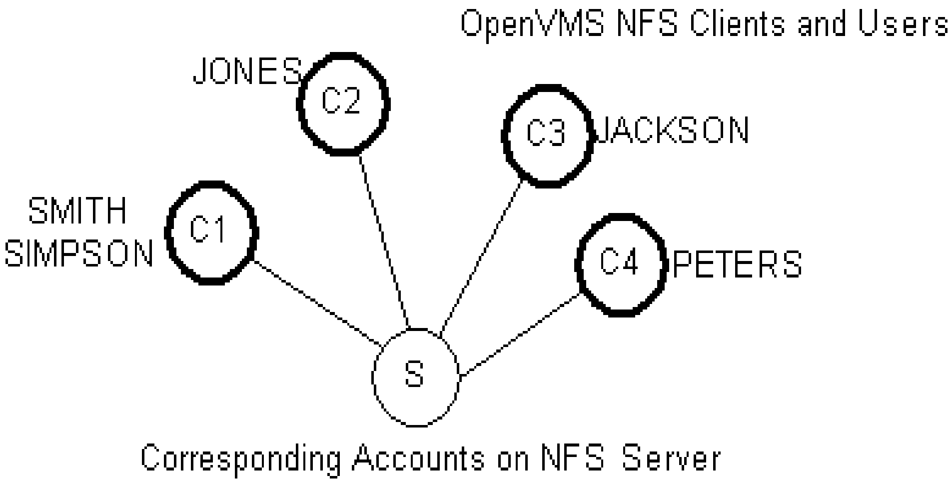
The management concepts discussed in this chapter include:

- The NFS client-server concept
- User and file protection
- Filename and file version mapping

Client-Server

The Client provides access to NFS-served filesystems while resolving differences in user space and file access between the two systems. Consider the scenario in Figure 12-1 where separate clients use a single NFS server.

Figure 12-1   Client-Server Concept



The indicated users for OpenVMS clients C1, C2, C3 and C4 need access to corresponding accounts on server S. If client users want group privileges to server files, the client system manager must create a group identity for the client users that maps to a group identity on S.

For example, SMITH and SIMPSON on C1 must have access to the `smith` and `simpson` accounts on S. If they also want group access to files on S, the client system manager must give them group identity (say, ACCOUNTING) on C1, even though they may not be part of the same user group on C1.

User and File Protection

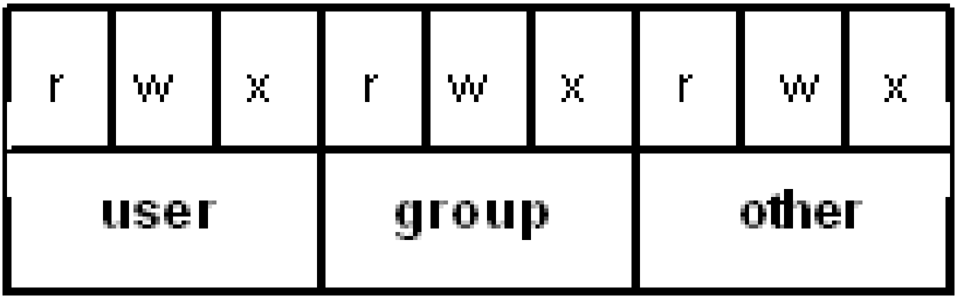
User and file protection are different in NFS and OpenVMS. Because of this, TCPware has to map user and file protection between the systems.

In order for the client to perform a server operation, two things apply:

1	The server must authorize the operation based on what its account can do following NFS/UNIX rules.
2	The client does its own user and file protection checking following OpenVMS rules.

The server has ultimate authority as to whether it should let the client perform the operation and may deny access based on NFS rules. Figure 12-2 shows this.

Figure 12-2 User and File Protection Checking



So that your client users can have access to server files:

1	Make sure the server system manager has the name of your client host in its export database. On many UNIX systems, this database is in the `/etc/exports` file; on hosts running TCPware Server, it is the EXPORT database.
2	Register each of your local users as having individual access to the appropriate server accounts. Do this by maintaining the PROXY database.
3	Register each of your local users as having the same group access to files as user groups on the server. Do this by maintaining the GROUP database.

The Client protects files and checks file access on the server using the following criteria:

- User and group identification of whoever requests access to a file
- Owner of the file
- Type of file access the user can have
- Special user privileges

The following sections cover these criteria more fully.

User and Group Identification

One way to protect a file is to check the identity of the user requesting access. If the server identifies that the user has access to the file, the server grants access to it.

NFS User Identification. NFS uses UNIX semantics. These consist of a User ID (UID), Group ID (GID), and GID list. A user has a unique UID, belongs to a primary group, and can be a member of a limited number of other groups.

All NFS hosts must share the same user space so that a user has the same identity on all systems. Because an account with a single UID on the server can belong to many groups (can have multiple

GIDs), you must associate a list of groups with that account.

Most UNIX servers have `/etc/passwd` and `/etc/group` files that maintain UID, GID, and group list information. The `/etc/passwd` file includes the account's login name, password, UID, and GID. The `/etc/group` file includes group names and their associated GID numbers and list of users. Each user can have a group list of up to 16 GIDs.

Parts of sample `/etc/passwd` and `/etc/group` files appear in Example 22-1 and Example 22-2 respectively.

Example 12-1 Sample `/etc/passwd` File

```
nobody:Nologin:-2:-2:anonymous NFS user:/:/bin/date
ris:Nologin:11:11:Remote Installation Services
Account:/usr/adm/ris:/bin/sh
daemon:*:1:1:Mr Background:/:
sys:PASSWORD HERE:2:3:Mr Kernel:/usr/sys:
bin:PASSWORD HERE:3:4:Mr Binary:/bin:
root:PASSWORD HERE:0:1:supervisor:/
edwards:PASSWORD HERE:100:/usr/users/edwards
```

Example 12-2 Sample `/etc/group` File

```
login:*:15:joe2
other:*:20:
accounting:*:10:edwards,root
testing:*:11:edwards,root
```

OpenVMS User Identification. In OpenVMS, a user has a unique user ID code (UIC) in the format [*group*,*member*], where *group* and *member* are alphanumeric, or in the format *USERNAME*, which is the *member* part of the UIC. For example, a UIC can be [306,210], [GROUP1,JONES], or just JONES.

You can also identify groups of OpenVMS users through general or system-defined rights identifiers. An example is the ACCOUNTING identifier that gives all users in the accounting department the same access rights to files. The OpenVMS system manager defines the general identifiers in the system rights database using the AUTHORIZE utility.

Table 12-1 reviews the differences between NFS and OpenVMS system user identification.

User Identification Mapping. User identification mapping between client and server is straightforward. Because an NFS account has the same UID across multiple groups, the Client maps UIDs directly to OpenVMS UICs. If the Client finds an appropriate mapping entry in the

PROXY database, the local user has access to the server account.

Table 12-1 User Identification in NFS and OpenVMS

NFS user identification...	Compared to OpenVMS user identification...
User ID (UID), Group ID (GID): identified as: *uid* *gid* as in: 100 15	User Identification Code (UIC): GROUP number MEMBER number identified as: [*group*,*member*] as in: [306,210]
GID List: as in: 16,17,18	Rights Identifier: as in: ACCOUNTING

Group Identification Mapping. Group identification mapping occurs through a special GROUP database because of the difference between the NFS and OpenVMS group concept. This database ensures that the group privileges in OpenVMS more accurately reflect the file group privileges on the server side.

Although OpenVMS users may be in the same OpenVMS group, they must take into account that their corresponding NFS server accounts may be in different groups. NFS accounts in the same group should allow group access to their corresponding users in OpenVMS, even though the latter may not belong to the same UIC-based group.

You must populate the GROUP database, as well as the rights identifiers list in OpenVMS. Entries in the GROUP database map NFS group numbers to assigned OpenVMS groups. The mappings are either to wildcarded OpenVMS group entries, such as [1000,*] (which means "group 1000, any member"), or to rights identifiers, such as ACCOUNTING.

Table 12-2 reviews how the Client handles user identification mapping.

Table 12-2 User Identification Mapping from Client to Server

OpenVMS user identification...	Maps using...	For NFS authorization...
UIC	PROXY database	UID, GID
UIC, Rights Database	GROUP database	GID List

File Ownership and Protection

NFSFile Ownership and Protection. Each NFS file has an owner and access restrictions (file protection) for various classes of users. File ownership and protection are file attributes.

Each NFS file has a UID and GID. When you create a new file, the NFS system:

- Sets the file’s owner UID to the effective UID of the creating process.

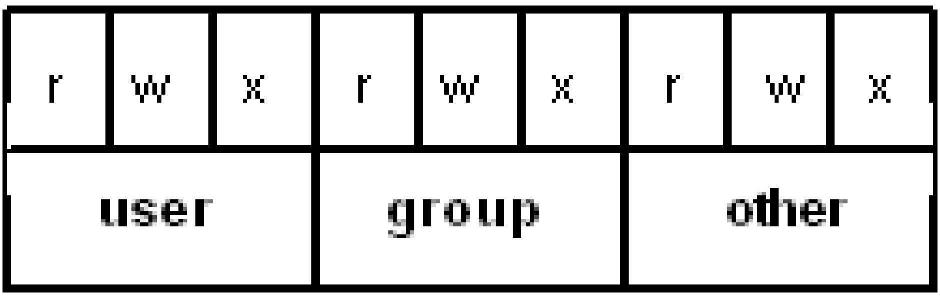
- Bases the file's owner GID on the `set-gid` bit in the file's parent directory:

If on	the owner GID becomes that of the parent directory.
If off	the owner GID becomes the effective GID of the creating process.

NFS systems use a protection mask scheme for file protection. The NFS file protection categories are part of the file's mode attribute and are `user`, `group`, and `other`, each having read (`r`), write (`w`) or execute (`x`) access.

NFS systems arrange the protection masks as in Figure 12-3.

Figure 12-3 NFS System Protection Masks



You can see the protection mask when you issue an `ls -l` command on the UNIX system server, as in the following example:

```
> ls -l
```

total 13949

```
-rwxr-x--x 1 smith 13330Jan 15 17:31 book
-rwxr-x--- 1 smith   44 Jan 15 17:31 games.com
drwxr-x--- 2 smith  512 Jan 15 17:38 Work
drwxr-x--- 1 smith   63 Jan 15 17:31 MARKET.rpts
```

In the example, the `book` file grants read (`r`), write (`w`), and execute (`x`) access to the file's `user`; `r` and `x` access to the `group`; and `x` access to the `other` category of users.

The lines beginning with `d` indicate directories. None of the files other than `book` provide access for the `other` user category, nor do any of the directories.

OpenVMS File Ownership and Protection. You own a file in OpenVMS if your UIC and the file's owner UIC are the same. When you create a new file, its owner is:

- The owner of the previous version of the file if you have rights to that owner
- The parent directory's owner if you have rights to that owner
- Your UIC

Each OpenVMS file has a protection mask that defines:

- The categories assigned to the file
- The types of access granted to each category

The four categories of OpenVMS file access 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).

OpenVMS arranges the protection masks as in Figure 12-4.

Figure 12-4 OpenVMS Protection Masks

In the following example, the SYSTEM and OWNER categories both have read (R), write (W), execute (E), and delete (D) access to the file. However, the GROUP category only has R and E access, and the WORLD category has no access at all:

```
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=<NO ACCESS>
```

File Ownership Mapping. Table 12-3 shows how the Client maps file ownership between server and client.

Table 12-3 File Ownership Mapping

NFS file attribute...	Maps using...	For OpenVMS file attribute...
UID, GID	PROXY database	Owner UIC
GID List	GROUP database	(special group handling)

File Protection Mapping. File protection mapping from server to client is slightly different than mapping from client back to server. Both map the access privileges for non-group file access categories to the corresponding privileges on the other system. However, you must establish group access through the GROUP database.

The Client handles file protection mapping from server to client as in Table 12-4.

Table 12-4 File Protection mapping from Server to Client

NFS category...	In OpenVMS is...	With NFS type...	In OpenVMS is...
user	OWNER/ SYSTEM	r	R
		w	W
		x	E
			D (unless ADF denies) [1]
group	GROUP	r	R (if GROUP database allows) [2]
		w	W (if GROUP database allows) [2]
		x	E (if GROUP database allows) [2]
			D (unless ADF denies) [1]
other	WORLD	r	R
		w	W
		x	E
			D (unless ADF denies) [1]

1 The Client allows delete (D) access only if a special attributes data file (ADF) the Client may create (and associates with the file) does not explicitly deny file deletion.

2 If a GROUP entry that maps to a rights identifier (such as ACCOUNTING) exists, the Client ignores the group protection mapping as given and uses the protection scheme in the special Access Control List (ACL) it creates instead. If a GROUP entry that maps to other than a rights identifier (such as a wildcarded group reference like [1000,*]) exists, the Client honors the group protection mapping as given.

Note! The Client honors the file protection scheme in the special, invisible ACL it creates for the file, and not in any other regular ACL.

The Client also handles file protection mapping from client back to server (such as when you create a file or change its attributes in OpenVMS), as in Table 12-5.

Table 12-5 File Protection Mapping from Client to Server

OpenVMS category...	In NFS is...	With OpenVMS type...	In NFS is...
SYSTEM	(not mapped)		
OWNER	user	R	r
		W	w
		E	x
		D	(not mapped)
GROUP	group	R	r
		W	w
		E	x
		D	(not mapped)
WORLD	other	R	r
		W	w
		E	x
		D	(not mapped)

Special Users and Privileges

Systems have users (or privileges given to users) that OpenVMS treats specially when checking access.

OpenVMS provides SYSPRV privilege (which gives access to the SYSTEM category), BYPASS privilege (which bypasses all checking, giving all access), and READALL privilege (which provides a user at least READ and CONTROL access).

So that the NFS server can grant these privileges, the user must have `superuser` access on the server. The `superuser` usually acquires UID=0.

The Client places undefined users by default in the `nobody` category, which provides a set of minimum access privileges. UID=-2 becomes user `nobody` and GID=-2 becomes group `nobody`.

Filename Mapping

For specific rules for mapping filenames between client and server, see Appendix A, *NFS-to-OpenVMS Filename Mapping*.

File Version Mapping

File version mapping can get rather complex due to the difference in behavior between OpenVMS and NFS systems. The general rule is that OpenVMS supports file versions; many NFS systems do not, and simply overwrite files on upgrading them. However, the TCPware Client does preserve file versions on the server (unless you use the NFSMOUNT /NOVERSION qualifier to limit file versions to one).

The Client still preserves an unversioned file on the server, which it hard-links to the highest (not necessarily most recent) version of the file every time it is upgraded.

In OpenVMS you could find the following DIRECTORY output:

Directory NFS4:[000000]
FILE-A.TXT;2 FILE-A.TXT;1 FILE-B.TXT;3 FILE-C.TXT;1
Total of 4 files.

The corresponding `ls` output on a UNIX NFS server would have the same files as follows:

```
total 6
174771-rwxr-x---2 root5 Jun2  11:36file-a.txt
174768-rwxr-x---1 root2 Jun2  11:35file-a.txt;1
174771-rwxr-x---2 root5 Jun2  11:36file-a.txt;2
174769-rwxr-x---2 root2 Jun2  11:36file-b.txt
174769-rwxr-x---2 root2 Jun2  11:36file-b.txt;3
174770-rwxr-x---1 root2 Jun2  11:36file-c.txt
```

Table 12-6 shows the file version rules when translating files from OpenVMS to NFS. Table 12-7 shows the file version rules when translating files from NFS to OpenVMS.

Table 12-6 OpenVMS-to-NFS Filename Translation Rules

Rule	What Happens to Filenames from OpenVMS to NFS...
1	An initial version of a file gets no version number: **`FOOBAR.TXT;1`** becomes **`foobar.txt`** EXCEPTION: A file explicitly created as version 1 when a higher version already exists, which creates an explicit **`foobar.txt;1`**.
2	An upgraded file is linked with the unversioned file, and if the previous version was unversioned, it gets a version number: **`FOOBAR.TXT;2`** becomes **`foobar.txt`** (with a hard link to **`foobar.txt;2`**) **`FOOBAR.TXT;1`** becomes **`foobar.txt;1`** This rule also applies if using NFSMOUNT /NOVERSION and upgrading a file that already has a version in NFS, or creating one with an explicit version.

Table 12-6 OpenVMS-to-NFS Filename Translation Rules (Continued)

Rule	What Happens to Filenames from OpenVMS to NFS...
3	If using NFSMOUNT /NOVERSION and upgrading a file that shows only as unversioned in NFS, the file is overwritten and remains unversioned: **`FOOBAR.TXT;1`** becomes **`foobar.txt`** (with **`foobar.txt;1`** purged) EXCEPTION: An attributes data file (ADF) specifies a version limit other than one, or an explicit version upgrade is specified.

Table 12-7 NFS-to-OpenVMS Filename Translation Rules

Rule	What Happens to Filenames from OpenVMS to NFS...
1	An unversioned file gets a version number preceded by a semicolon: **foobar.txt** becomes **FOOBAR.TXT;1**
2	If a filename does not include a file extension dot (.), it acquires one before the version number semicolon: **foobar** becomes **FOOBAR.;1**
3	After being translated, the file will not show up in the OpenVMS listing if its version number is greater than 32767.

Filesystem Mounting

The Client links authorized (exportable) remote NFS filesystems to your OpenVMS system by mounting them (making them available) on a file structure you specify.

OpenVMS arranges file storage the user can access in directory trees. OpenVMS roots each tree (at the top) at an NFS device (such as NFS1:). The format of an NFS device is the following, where *n* is a number from 1 to 9999, is **`NFS`*`n`*`:`**.

If you specify **`NFS0:`**, the Client uses the template device and increments the *n* number by one for each new mount on the client host. For example, if someone mounts a filesystem on your host's NFS5: device and you specify a mount on NFS0:, the next mount is on NFS1: (or the next available device). The Client uses the template device only when you specify NFS0: or omit the mount point specification entirely.

The mount point is both the point on the remote directory tree to be mounted and the point on the local directory tree where the Client "attaches" the remote filesystem. A mount point can be at any level. OpenVMS's Record Management Services (RMS) lets you specify eight directory levels in addition to the master file directory (MFD or [000000]).

The most common Client mount point is the MFD of the NFS device (NFS*n*:[000000]). Lower level mounts create virtual directories. For example, if you specify a mount point NFS1:[A.B], the Client creates virtual directories A and B on NFS1. These directories disappear once you dismount the filesystem.

NFS mounts are node-specific and not cluster-wide. Other nodes in the cluster cannot access these files. This has implications for printing files and running batch jobs and programs in a cluster. (See the next section.)

Cluster Environments

NFS is not VMS clustering. VMSclusters use RMS file locking that is more tightly coupled than the NFS advisory file level locking mechanism. In NFS, cluster-wide programs that store or exchange file IDs are unlikely to function properly. The NFS device is not available cluster-wide and the same filesystem mounted on different nodes has different file IDs.

The best NFS strategy is to allow only one NFS client system to write to the server files. If you need multiple clients to write to the same file, use the Network Lock Manager (NLM) by specifying NFSMOUNT /LOCK. Also ensure that all systems (client and server) use the NLM to coordinate file access of the participating systems. NLM provides advisory locking, not mandatory locking as with VMSclusters.

The fact that the Client mounts filesystems only on the local OpenVMS cluster node has implications for printing files and running batch jobs and programs in a cluster environment. In printing a file, make sure that you set up the printer on the node on which you mount the NFS filesystem. Also make sure that no one remounted the NFS filesystem during the print operation. Otherwise the print operation fails because the Client changes the file ID with each mount (the printing system keeps track of jobs using file IDs). The same applies to batch jobs involving NFS files.

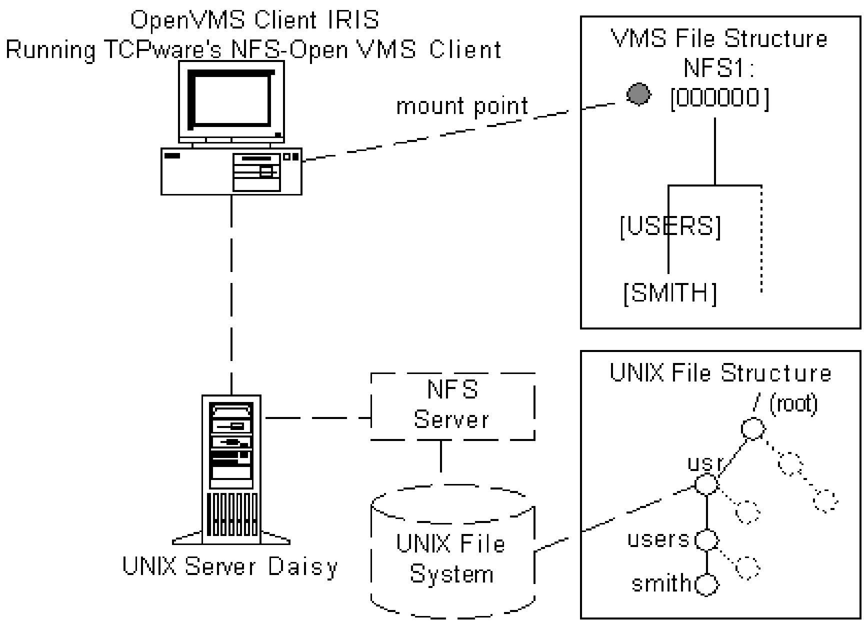
To print files in or submit batch jobs for mounted NFS filesystems across the cluster, first copy the files to a "standard" non-NFS disk which you can access cluster-wide.

For details on network file locking and its implications for both client and server users, see Chapter 13, *NFS-OpenVMS Server Management*, *Network File Locking*. Also see *Network File Locking* in this chapter.

Mount Example

Figure 12-5 shows an example of an exported UNIX filesystem mounted in OpenVMS.

Figure 12-5   Mounting a UNIX Filesystem on the OpenVMS Client



In the figure, an OpenVMS user on host IRIS needs access to the `/usr/users/smith` subdirectory on UNIX system server Daisy. Other IRIS users may need access to subdirectories below `/usr`.

Using the NFSMOUNT command at the DCL prompt, IRIS's system manager mounts `/usr` on IRIS's NFS1: device, where the files are now "attached" to NFS1:[000000]. The Client creates two virtual directory levels ([USERS.SMITH]) below NFS1:[000000]. If the user wants access to files in `/usr/users/smith`, the way to access them is through NFS1:[USERS.SMITH].

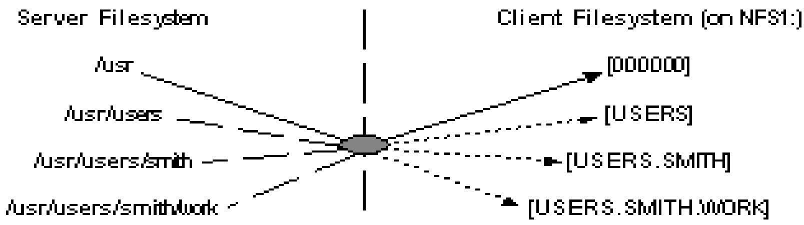
Had the system manager mounted `/usr/users/smith` instead of just `/usr` on the same mount point, `/usr/users/smith` would be equivalent to NFS1:[000000]. However, the user would then be excluded from any directory level above `/smith` (`/usr` and `/users`).

Mount Flexibility

The flexibility of the Client's mount capabilities appears in Figure 12-6. The NFS filesystem is on the left and the corresponding OpenVMS filesystem is on the right. Each row is a directory level. The solid line pointing from `/usr` to [000000] indicates the mount scenario indicated in Figure 12-5, where the user mounted the `/usr` filesystem on NFS1:[000000].

The dotted lines indicate that you can essentially bend the arrow anywhere around the central pivot point so that you can mount any server filesystem level on any client filesystem level. You can mount a low server directory on a high client mount point, a high directory on a low mount point, a low directory on a low mount point, and so on.

Figure 12-6   Possible Client Mount Scenarios



You can even mount a file on a file, such as `/usr/users/smith/junk.txt` on NFS1:[USERS.SMITH]JUNK.TXT. However, mounting a file limits the user to accessing that file only. This makes new file creation impossible since the Client cannot "go up" the directory tree to get the necessary attributes to create a new file.

You can only access server resources from the mount point on down. For example, if you use the following NFSMOUNT command, you can access all available resources in Daisy's `/usr/user/smith` directory mounted locally as NFS1:[USERS.SMITH], but you cannot back up to NFS1:[USERS] and expect to access resources in Daisy's `/usr` directory that are not in the `/users` subdirectory:

```
$ NFSMOUNT DAISY "/usr/users/smith" NFS1:[USERS.SMITH]
```

You can use NFSMOUNT command options for certain mount strategies for your specific implementation, such as automounting and background mounting.

For details, see *Mount Strategies*.

Mount Commands

The Client commands related to file mounting appear in Table 12-8.

Table 12-8   Client Mount Commands

Command	Description
$ NFSMOUNT	Mounts a remote NFS directory on a local mount point; similar to UNIX systems' `/etc/mount` file
$ NFSDISMOUNT	Dismounts a remote NFS directory from a local mount point; similar to UNIX systems' `/etc/unmount` file
$ NFSMOUNT /CONFIG	Mounts a remote NFS directory on a local mount point based on information in a configuration file
$ NFSMOUNT /SHOW	Shows the mount information for the local mount point

Perform mount commands at the DCL level. The NFS mount command format is:

```
NFSMOUNT server "pathname" [mountpoint [logical]]
```

server	is the name of the remote NFS server.
"*pathname*"	is the server's exported pathname (enclosed in quotation marks).
mountpoint	is the optional NFS device (and directory path, if necessary) where the mount occurs on the OpenVMS host. If you do not specify the mount point, the default is NFS0:[000000], the MFD mount on the template device, as discussed earlier.
logical	is the optional logical name associated with the *mountpoint*.

The following command mounts Daisy's `/usr/users` filesystem on the NFS1:[000000] mount point:

```
$ NFSMOUNT DAISY "/USR/USERS" NFS0: DAISY$USR_USERS
```

The Client immediately creates the NFS1: mount point based on the NFS0: template specification. The mount point also gets the (optional) logical name DAISY$USR_USERS.

The NFS dismount command format is:

```
$ NFSDISMOUNT mountpoint
```

where *mountpoint* is the mount point or logical name, as defined above.

Mount Strategies discusses the various mount and dismount options. For details on the mount and dismount commands, see *Client Commands*.

Symbolic Links

The Client supports symbolically linked (known as "soft-linked") files on UNIX servers. This support preserves existing symbolic links when you back up your client filesystems and restore them on these servers.

The Client does not "chase" symbolic links. If you open a soft-linked file in OpenVMS, it displays the pathname of the linked file, unlike UNIX systems that display the actual content of the linked file.

Client Auditing

The Client supports OpenVMS security auditing that keeps track of security events users can cause and issues security alarms for these events.

See Compaq's *Guide to VMS System Security* for details.

Mount Strategies

The Client offers several ways to mount a filesystem:

- Regular mount
- Shared mount
- Automount
- Background mount
- Overmount
- Occluded mount
- Mount using network file locking

Regular

The following shows a sample confirmation message given when mounting SIGMA's `/usr` directory on an NFS0: template device:

```
$ NFSMOUNT SIGMA "/usr" NFS0:
%NFSMOUNT-S-MOUNTED, /usr mounted on NFS101:[000000]
```

If you use the same command again, the Client creates a new mount point (such as `_NFS102:[000000]`).

There are other options you can set using command qualifiers during a regular mount, such as setting SYSGEN parameters and cache timeout.

These options appear in *Other Mount Options*. For the mount qualifiers, see the NFSMOUNT command in *Client Commands*.

Shared

The Client supports shared mounting through the use of the /SHARE qualifier to the NFSMOUNT command. The format of a shared mount request is as follows:

```
$ NFSMOUNT BART "/ENG" NFS1: BARTMOUNT
$ NFSMOUNT/SHARE BART "/ENG" NFS1:
```

The first mount request creates the NFS1: device, and puts the BARTMOUNT logical in the system logical name table. The mount count is set to one. The second (shared) mount request, which includes the same mount information, increments the mount count by one. Unless you specify the /SHARE qualifier and the same hostname, path, and device/mount point for the second mount request as for the first, the second mount request is considered a new mount request and not a shared mount request.

Logical names go into the SYSTEM logical name table by default. A /SHARE mount, however, puts the logical name in the JOB logical name table. In this way the logical name is only available to processes of the job mounting the device.

The mount count increments and decrements under the following circumstances, instead of being automatically set to zero:

- With an initial SYSTEM or GROUP mount
- With a DCL MOUNT /SHARE or TCPware NFSMOUNT /SHARE command that completes without an error (the exception being an overmount, where the previous mount point is dismounted, in which case it may even be decremented if the previous mount point was shared)

In this way, if the main process of the job logs out, the job mount is deallocated, and the volume mount count decrements by one (and if zero, dismounts the device). OpenVMS handles *dismounting* differently based on whether you use TCPware's NFSDISMOUNT or the DCL DISMOUNT command:

If using...	**Then...**
TCPware's NFSDISMOUNT	The NFS Ancillary Control Process (NFSACP) dismounts one or more (in the case when using the /ALL qualifier) 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 than one mount point exists, the mount point is dismounted along with any specifically shared mounts.
DCL DISMOUNT	OpenVMS checks for JOB mounts first. If a JOB mount for the specified device exists, the JOB mount is dismounted, the associated logical (if any) is deallocated, and the mount count is decremented by one. If no JOB mount exists, OpenVMS checks for SYSTEM and GROUP mounts. If one of these exists and the user has the required privilege (SYSNAM or GRPNAM), the SYSTEM or GROUP mount is dismounted, any associated logical name is deallocated, and the mount count is decremented by one. No mount points are dismounted until the mount point hits zero. If the user does not have the required SYSNAM privilege, the error message `SYSTEM-F-NO-SYSNAM, operation requires SYSNAM privilege` is returned, with a similar message for the GRPNAM privilege.

Consider the mount counts in the mount/dismount sequence shown in Table 12-9.

Table 12-9 Mounting and Dismounting a Shared Mount

	Command...	**Creates mount count...**
1	NFSMOUNT BART "/ENG" NFS1:[A]	1 (system: increment)
2	NFSMOUNT/SHARE BART "/ENG" NFS1:[A]	2 (shared: increment)

Table 12-9 Mounting and Dismounting a Shared Mount

	Command...	**Creates mount count...**
3	MOUNT/SHARE NFS1:	3 (shared: increment)
4	NFSMOUNT MARGE "/TEST" NFS1:[B]	3 (system again: do not increment)
5	NFSDISMOUNT NFS1:[A]	2 (remove #1: do not decrement, and remove #2: decrement)
6	DISMOUNT NFS1:	1 (remove #3: decrement)
7	DISMOUNT NFS1:	0 (remove #4: decrement)

Note! That the mount count in #5 is not decremented because the first NFSMOUNT is a system mount. The original mount for BART "/ENG" on NFS1:[A], along with its shared mount, is dismounted. #6 and #7 further decrement the mount count to zero.

Automounting

Use the /AUTOMOUNT qualifier to automount a filesystem, which automatically and transparently mounts a remote server path as soon as the Client obtains the pathname. You can specify an inactivity period (the default is five minutes, seconds being rounded to the nearest minute), after which the Client automatically dismounts the path.

In this example, the Client mounts SIGMA's `/usr` filesystem when it references the pathname, keeping the path mounted until an inactive period of 10 minutes, after which time it unmounts the filesystem (subsequent references causing the Client to remount it):

```
$ NFSMOUNT SIGMA "/usr" NFS0: /AUTOMOUNT=(INACTIVITY=00:10:00)
```

Background

Use the /BACKGROUND qualifier to mount a filesystem in background mode, where the Client tries to mount a filesystem at least once. If the first try fails, the Client informs you of the failure and tries again every 30 seconds (by default). Unless you set it otherwise, the Client stops retrying after 10 attempts.

In this example, the Client tries to mount the `/usr` filesystem; if it cannot, it waits one minute and retries the connection up to 20 times:

```
$ NFSMOUNT SIGMA "/usr" NFS0: /BACKGROUND=(DELAY=00:01:00,RETRY=20)
```

Overmounting

Use the /FORCE qualifier to overmount a filesystem, where you mount another path on an already existing mount point. The Client dismounts the original filesystem and replaces it with the new one. (If you omit /FORCE, the message `overmounting requires the use of /FORCE` appears.) Mounting a higher or lower directory level in a previously used path also causes an overmount, as in the following example:

```
$ NFSMOUNT SIGMA "/usr" NFS1:[USERS.MNT]
%NFSMOUNT-S-MOUNTED, /usr mounted on _NFS1:[USERS.MNT]

$ NFSMOUNT SIGMA "/usr/users" NFS1:[USERS.MNT] /FORCE
%NFSMOUNT-S-REMOUNTED, _NFS1:[USERS.MNT] remounted as /usr/users on
SIGMA
```

Occluded

Use the /FORCE qualifier for an occluded mount, where you mount a filesystem onto a client mount point that is higher or lower in the directory structure than an existing, active mount. (If you omit /FORCE, the message `occlusion requires the use of /FORCE` appears.) No dismounting occurs and both mounts are visible. However, the Client occludes (hides from view) the subdirectory (or subdirectories) added to or dropped from the original mount specification when you perform a directory listing.

In the following example, the mount point specification moves up one subdirectory. If you enter the NFSMOUNT /SHOW command, both mounts are visible. However, if you do a directory listing on NFS2:[USERS.SMITH], the [MNT] directory is no longer visible. To make the directory visible again, dismount NFS2:[USERS.SMITH].

```
$ NFSMOUNT SIGMA "/usr" NFS2:[USERS.SMITH.MNT]
%NFSMOUNT-S-MOUNTED, /usr mounted on _NFS2:[USERS.SMITH.MNT]
$ NFSMOUNT SIGMA "/usr" NFS2:[USERS.SMITH] /FORCE
%NFSMOUNT-S-MOUNTED, /usr mounted on _NFS2:[USERS.SMITH]
-TCPWARE-I-OCCLUDED, previous contents of _NFS2:[USERS.SMITH]
occluded
```

Network File Locking

Use the NFSMOUNT /LOCK command to enable network file locking during an NFS mount. The NLM applies the lock to any file you create or to which you request exclusive access in the specified filesystem. The locks are on entire files only and not on file regions or byte ranges. Here is a typical example:

```
$ NFSMOUNT SIGMA "/usr" NFS0: /LOCK
```

Other Mount Options

This section identifies other mount options you can set.

Auto-converting Text Files

By default, the Client automatically converts newly created text files of variable-length, implied carriage return control (VAR-CR) format to STREAM-LF format. This is appropriate for UNIX system servers, where the files show up as `stream_lf`. For a PC system server, however, use the NFSMOUNT /CONVERT=STREAM_CRLF option to do a carriage-return-line-feed conversion for the mount point. Converted files will show up on the server as stream files, as do files that do not have attributes data files (ADFs) associated with them (see the next section).

Some OpenVMS applications require that certain files remain in VAR-CR format on the client (such as with TCPware's NFS-OpenVMS Server). You can retain the VAR-CR format by specifying the /NOCONVERT qualifier during a mount. For example:

```
$ NFSMOUNT SIGMA "/usr" NFS0: /NOCONVERT
```

Attributes Data Files

Attributes data files (ADFs) are special companion files the Client maintains on a non-VMS server to preserve attribute information the server would not normally recognize.

The Client maintains regular and default ADFs for files such that:

1	If a regular ADF exists, the Client uses the attributes from that file.
2	If a default ADF exists, the Client uses the attributes from that file.
3	If no ADF exists, the file must be STREAM-LF.

The Client may create a regular ADF for a file in response to a write attributes operation that sets an OpenVMS attribute that NFS cannot normally maintain. For example, a SET FILE /NOBACKUP command would cause the Client to create an ADF, since NFS has no concept of this OpenVMS attribute.

Default ADFs minimize the number of regular ADFs, since one default ADF can serve all files of a particular type. The Client provides default ADFs for files with .EXE, .HLB, .MLB, .OBJ, .OLB, .STB, and .TLB extensions. The Client does not provide ADFs for files with the .TXT and .C extensions, since most of these are STREAM-LF.

For example, TCPWARE:EXE.ADF is the default ADF for all .EXE type files. When you create .EXE files (or if they exist on the server), the record attributes from the single default ADF are probably enough to define each file. Each file does not need its own regular ADF.

Note! The Client uses only certain record attributes and file characteristics in default ADFs. It uses the 32-byte ATR$C_RECATTR attributes *other than* the FATL_HIBLK, FATL_EFBLK, FAT$W_FFBYTE, and FAT$W_VERSION fields, and uses four-byte ATR$C_UCHAR attributes *other than* the FCH$M_DIRECTORY and FCH$M_CONFIG bits. All other information stored in an ADF is ignored for default ADFs. For additional details on these file attributes, see Compaq's *OpenVMS I/O User's Reference Manual*, the *ACP-QIO Interface* chapter.

When a user creates a file on the client, the Client only creates a "regular" ADF for it if the default ADF attributes or default attributes do not match.

You can create customized ADFs for special applications. To do so:

1	On the client, create a special application file that creates an ADF on the server.

2	Suppose that application file is TEST.GAF. On the server, check the listing for the data file, along with its ADF (`.$ADF$test.gaf;1`):
	
	`>` **`ls -a`**
	`.`
	`..`
	`.$ADF$test.gaf;1`
	`test.gaf`

3	On the server, copy the ADF file to a newly created default ADF file on the client: `>` **`cp .ADFtest.gaf1 gaf.adf`** Note the backslashes (\) required to recognize the UNIX system non-standard `$` symbol and the `;` symbol required to specify a version number.
4	On the client, copy the new default ADF file to the TCPWARE_COMMON:[TCPWARE] directory: `$` **`COPY GAF.ADF TCPWARE_COMMON:[TCPWARE]`**
5	Dismount all the NFS volumes and remount them again. This starts another NFS ancillary control process (ACP) so that the newly copied default ADF file can take effect.

You can also specify how you want ADFs used. The Client offers three options with the /ADF qualifier of the NFSMOUNT command:

CREATE	If ADFs exist on the server, the Client uses them, updates them, and creates them for new files. This setting is the default and recommended setting.
UPDATE	If ADFs exist on the server, the Client uses them and updates them, but does not create them for new files.
USE	If ADFs exist on the server, the Client uses them, but does not update them, nor does it create them for new files.

Note! Avoid using UPDATE and USE. The Client may still create ADFs in certain cases, such as when renaming files. Also, changing OpenVMS attributes for a hard-linked file can result in inconsistent OpenVMS attributes between the linked files.

You can also specify /NOADF. In this case, the Client treats all files as STREAM-LF unless a default ADF matches and it can use it. Note that this mode of operation is only appropriate for read-only filesystems, since the Client cannot adequately handle application-created files when /NOADF is in effect.

Cache Timeout

Use the /CACHE_TIMEOUT qualifier to set the caching timeout period for the mount point. For example:

```
$ NFSMOUNT /CACHE_TIMEOUT=(DIRECTORY=::15,ATTRIBUTE=::10)
```

```
The DIRECTORY timer specifies the time (in delta time) the Client
waits between rereading a directory's status or contents. The default
is ::30 (30 seconds). The ATTRIBUTE timer specifies the time the
Client waits between rereading a file's attributes from the server.
The default is ::15 (15 seconds).
```

Read/Write Transfer Size

Use the /DATA qualifier to specify the largest amount of NFS data you want to read (receive) or write (transmit) in a single network operation. For example:

```
$ NFSMOUNT /DATA=(1024,1024)
```

The first value is the read value and the second is the write value. Most servers let you read or write 8192 bytes (the maximum and default setting). However, some may require less. The minimum you can specify is 512 bytes.

If you eliminate the parentheses and specify only one value, this serves for both the read and write value. However, if the NFS server requests a smaller transfer size than the one you set, the server's requested value will override yours.

Default User

Use the /USER qualifier to specify the default user to which you want to map unknown UIDs. For example:

```
$ NFSMOUNT /USER=SMITH
```

The Client tries to use the USER account. If not found, the DECNET account becomes the default. If the DECNET account is not found, the [200,200] account becomes the default.

Default UIDs and GIDs

Use the /UID and /GID qualifiers to specify the default UID and GID. The Client uses the default UID and GID if there is no mapping for the requesting user in the PROXY database. Usually the default UID is -2 and default GID is -2. For example:

```
$ NFSMOUNT /UID=100 /GID=15
```

Limiting File Versions

Use the /NOVERSION qualifier to enforce a limit of one version on a non-TCPware server file. This is a way of imposing an NFS file versioning scheme on OpenVMS files. (/VERSION, allowing multiple versions, is the default).

With /NOVERSION, unversioned files stay unversioned, and new files are unversioned along with any subsequent upgrades (which is consistent with most NFS servers). When higher versions already exist, the number of versions cannot grow beyond the current number, so that the lowest version is purged on each upgrade.

For example, with /NOVERSION in effect, if you start with FILE.TXT;1 (which shows up as `file.txt` on the server) and you edit FILE.TXT, you have an overwritten FILE.TXT;1 on the client, and `file.txt` on the server. If you already have FILE.TXT;1 and FILE.TXT;2 and you edit FILE.TXT, you end up with FILE.TXT;3 and FILE.TXT;2 on the client, with version 1 purged. The server shows `file.txt`, `file.txt;2`, and `file.txt;3` (hard-linked to `file.txt`).

To prepare a directory for use with /NOVERSION, it may be best to purge and rename its files, as follows, being aware that purged files are lost forever and to back up your files whenever possible:

```
$ PURGE *.*; deletes old versions
$ RENAME *.*;* *.*;1; forces server to rename files to unversioned
$ SET FILE /VERSION=1 *.*; overrides existing ADFs
```

Superusers

Use the /SUPERUSER qualifier if you want to allow access to the superuser account UID on the server. For example:

```
$ NFSMOUNT /SUPERUSER=200
```

To enable `superuser` privilege, the server must allow access to the `superuser` and the OpenVMS user must have SYSPRV, BYPASS, or READALL privileges. Normally, the superuser UID is 0. The default is /NOSUPERUSER.

Mount Type

If you specify a logical name for the mount point, the Client creates a system logical name by default. This is equivalent to using the /SYSTEM qualifier of the NFSMOUNT command. If you specify the /GROUP qualifier, the Client considers the mount a group mount and places the logical name (if specified) in the group table. Both mounts are subject to a privilege check.

Server Type

Use the /SERVER_TYPE qualifier to set the server type to UNIX, TCPware, or IBM_VM. By default, the server type is UNIX or TCPware, depending on the server. For example:

```
$ NFSMOUNT /SERVER_TYPE=IBM_VM
```

The server types displayed in Table 12-10 are available with the Client.

Table 12-10 /SERVER_TYPE Qualifier Options

Option	Description
IBM_VM	IBM Virtual Machine (VM) machines
TCPWARE	OpenVMS systems running NFS-OpenVMS Server
UNIX	All UNIX system machines

TCPWARE Server Type

When mounting a filesystem served by TCPware's NFS-OpenVMS Server (Version 4.0 and later), either omit /SERVER_TYPE or specify /SERVER_TYPE=TCPWARE. If omitted, the Client determines the TCPWARE server type automatically. Note the following:

- The Client and Server map UICs to UIDs and GIDs. As long as system managers on each system maintain the PROXY databases properly, this saves having to maintain the same set of UICs on the client and server systems.
- The Client and Server use ACLs as is. This means that identifiers on the client and server systems must be the same to produce the desired results.

IBM_VM Server Type

IBM's VM NFS server partially supports requests to change file size. This means that:

- OpenVMS Files-11 ODS-2 is a block-oriented filesystem. Applications (and RMS) can only read and write 512-byte blocks. The Client uses ODS-2 file attributes to maintain information about the true last data byte of a file.

  To accommodate the IBM VM NFS server's inability to truncate a file to its real size (the Client normally truncates the file based on the end-of-file information), the Client stores the size information in the ADF for the file.
- With any access to the file from a non-TCPware NFS client or an application on the server, you may see garbage beyond the true end of the data. (This garbage data exists because of the block nature of ODS-2 and the server's inability to truncate the file to its real size.)
- With a file stored on the IBM VM NFS server by a non-TCPware client or an application on the server itself, the ADF does not reflect any changes to the file. This can cause problems if a client later opens the file, expecting changes.

When mounting a filesystem on an IBM VM minidisk, you must specify certain IBM VM NFS server specific parameters in the pathname parameter of the mount command. For example, a mount to an IBM VM minidisk might be:

```
$ NFSMOUNT IBMVM "test2.191,rw,user=simpson,pass=bart,record=nl"
```

You may need to specify one or both of the following parameters:

- **record=**{**binary** | **nl** | **text**}

binary (default)	The IBM VM NFS server does not convert data to EBCDIC. This mode is most useful when storing data to which you do not have access from applications on the IBM system, or when transferring binary data.
nl	The IBM VM NFS server translates EBCDIC to ASCII (and vice versa). This mode is most useful when storing text files to which you have access from applications on the IBM system. Do not use it when you have access to or store binary data files.
text	The file conforms to the IBM VM CMS structure. Use of this parameter value is not generally recommended.

- **name=**{**trans** | **fold** | **mixed**}

`trans` (default)	Supports the widest range of filenames. The IBM VM NFS provides transparent mapping for filenames that contain invalid characters or are longer than CMS allows. However, the Client does not use this mapping if the filename (ignoring case) is valid on the CMS filesystem. Therefore, for short filenames, the mapping may not be transparent.
`fold`	Only supports filenames valid to the CMS filesystem and ignores case.
`mixed`	Like `name=fold` except that it preserves case.

For complete details on these server types, see the IBM *TCP/IP for VM: User's Guide*.

Retry Times

Use the /RETRIES qualifier to specify the maximum number of times the Client retransmits a Remote Procedure Call (RPC) request. For example:

```
$ NFSMOUNT /RETRIES=10
```

There is no maximum value you can specify. The default is zero (0) retries, where the Client retries the requests indefinitely.

Timeout Times

Use the /TIMEOUT qualifier to set the minimum timeout period (specified in OpenVMS delta time – see Table 12-12) for initial RPC request retransmissions.

The timeout period value should reflect the estimated typical round trip time for RPC requests. For slower speed links (such as over SLIP or WAN lines), use a larger value than the default of one second. For example, for a maximum read/write size of 8192 bytes (see the /DATA qualifier) over a 19,200-baud SLIP line, the absolute minimum timeout value is:

```
10240 bytes  * 8 bits per byte
-------------------------------------- =  4.27 seconds
19200 bits per second
```

The 10240 bytes are 8192 plus RPC overhead. Since 4.27 seconds is the absolute minimum, a more realistic value for this link is in the range of 15 to 30 seconds to allow for other traffic.

Volume Labels

Use the /LABEL qualifier to specify the volume label to use for the remote pathname during a mount. If you omit /LABEL, the Client uses a default label consisting of the first 12 characters of the combined *server*:*mountpoint* parameter. The Client applies the /LABEL qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device. If you perform a SHOW DEVICE NFS*n*: DCL command, you see only the first 12 characters of the volume label specified.

Cache Space

One of the options during a mount is the /PROCESSOR=UNIQUE qualifier setting. As a general rule, the larger the remote filesystem, the more likely you are to use this option.

With /PROCESS=UNIQUE, a new NFSACP process is created for each mount. This creates multiple address space, in which case the collective ACPs can accommodate much more cached information. The size of the cached information depends mostly on the number of NFS files the client recognizes by obtaining a file handle and creating a mapping to a file ID. This happens with any file or directory access.

Each NFSACP process can support up to 250 mounted filesystems. If one process is handling all mounts, there is only one address space to cache the information. The size of this address space depends on a number of system parameters such as VIRTUALPAGECNT, and process parameters such as the working set limits and paging file limits.

Disk Quotas

You can display quota information for the current user's mount by using the NFSMOUNT /SHOW command with the /QUOTA qualifier. The output displays block usage, soft limit (quota), hard limit, and grace period. Using the additional /FULL qualifier displays four additional values that are relevant to UNIX servers: file usage, quota, limit, and grace period.

You can use the additional /USER qualifier to request quotas for other than the current user. However, the NFSMOUNT required the /SUPERVISOR qualifier and SYSTEM, BYPASS, and READALL privileges. (The DCL command SHOW QUOTA also works in this case.)

The following shows sample output:

```
    $ NFSMOUNT /SHOW NFS2: /QUOTA /FULL

_NFS2:[000000]  mounted
  viola:/pctest
  Disk Quotas for user [SMITH]: (inactive)
  Blocks    Quota    Limit     Grace    Files    Quota    Limit Grace
  117355    500000   600000             0        0        0
  Transport                   UDP       Writing                 Enabled
  Read/write size       8192/8192       Write conversion       Disabled
  RPC timeout       0 00:00:01.00       ADF usage     USE,UPDATE,CREATE
  RPC retry limit               0       Fileids       Unique, Nolookups
  Attribute time    0 00:00:15.00       Server type             TCPware
  Directory time    0 00:00:30.00       Advisory Locking       Disabled
  Cache Validation    MODIFY TIME       Default user             [USER]
  Superuser                    No       Default UID,GID          100,15
```

Implementation

There are only minor differences between the way the Client and Files-11 ODS-2 handle files. For example, the Client:

- Does not determine the physical placement of files on the disk.

- Does not support the INDEXF.SYS file, which means that you cannot perform operations such as ANALYZE/VERIFY and BACKUP/IMAGE in OpenVMS.

Note! The NFS-OpenVMS Client is not supported in the POSIX environment.

Client Commands

Table12-11 shows the mount and dismount commands available at the DCL level in OpenVMS.

Table 12-11 Mount and Dismount Commands

DCL command	Description
$ NFSMOUNT	Mounts a remote NFS directory on a local mount point; similar to UNIX systems' `/etc/mount` file
$ NFSDISMOUNT	Dismounts a remote NFS directory from a local mount point; similar to UNIX systems' `/etc/unmount` file
$ NFSMOUNT /CONFIG	Mounts a remote NFS directory on a local mount point based on information in a configuration file
$ NFSMOUNT /SHOW	Shows the mount information for the local mount point

The mount and dismount commands use OpenVMS delta time for all time-related values.

The delta time syntax is:

```
dddd hh:mm:ss.cc
```

	Is the number of...
dddd	days (0-9999); if less than one day, specify zero (0); follow with a blank space
hh	hours (0-23)
mm	minutes (0-59) preceded by a colon (:)
ss	seconds (0-59) preceded by a colon (:)
cc	hundredths of a second (0-99) preceded by a period (.)

You can truncate a delta time on the right. You can omit fields in the time format as long as you include the punctuation that separates the fields. You must specify the days field even if you omit all time fields.

Table 12-12 shows some examples of delta time.

Table 12-12 Delta Time Examples

Delta time...	Can be truncated...	And means...
3	3	3 days from now
3 03:00:00	3 3	3 days and 3 hours from now
0 00:03:00	0 :3	3 minutes from now
0 00:00:03	0 ::3	3 seconds from now
0 00:00:00.03	0 ::.3	3 hundredths of a second from now

Troubleshooting

NFS-OpenVMS Client can produce messages for the NFSMOUNT and NFSDISMOUNT commands, and in OPCOM.

Access error messages help by entering **`HELP TCPWARE MESSAGES`** ***`[identifier]`***, or connect to web site **`http://www.process.com`** (select **`Customer Support`** followed by the **`Error Messages`** button).

NFSMOUNT

Mounts a remote NFS directory to a local mount point. The command is similar to the UNIX system `/etc/mount` command.

DCL Format

NFSMOUNT ***server "nfs-path" [mountpoint [logical]]***

Parameters

server

Name of the remote server, in domain name or IP address format.

"nfs-path"

Pathname (enclosed in quotation marks) on the remote server. The pathname must match an exported directory, subdirectory, or file of an exported filesystem on the server. (You can use the SHOW EXPORT command in the TCPware Network Control Utility (NETCU) to obtain a list of the exported directories.)

mountpoint

NFS device (and, optionally, directory tree) specification for the local mount point. If specified, this parameter must be in the format:

```
NFSn:[[dir.dir....]][filename]
```

The value *n* can range from 1 to 9999, and *dir* is a directory level (up to eight in addition to the [000000] directory). If you omit the *mountpoint* specification or specify **`NFS0:`**, the Client creates an `NFSn:[000000]` mount point, and increases *n* by one for each subsequent mount.

logical

Optional logical name associated with the volume. The Client defines the logical as follows:

- If you mount **NFS*n*:[000000]**   **NFS*n*:**
- If you mount **NFS*n*:[*dir.dir*]**   **NFS*n*:[*dir.dir.*]**

The extra dot after the last *dir* in the second definition allows for relative directory specifications. If you perform the following function:

```
SET DEFAULT logical:[subdir]
```

the full default definition becomes:

```
NFSn:[dir.dir.subdir]
```

The Client places the logical name in the SYSTEM logical name table unless you specify the /GROUP or /SHARE qualifier. The Client deletes the logical name from the SYSTEM table when you dismount the volume. The process must have SYSNAM privilege to mount a system mount point. Without SYSNAM or GRPNAM privilege, the user must specify /SHARE for a JOB mount.

Qualifiers

/ACP_PARAMS=(*[*BUFFER_LIMIT=*limit-value]*

[*,DUMP*]
[*,IO_DIRECT=*value]
[*,IO_BUFFERED=*value]
[*,MAX_WORKSET=*pages]
[*,PAGE_FILE=*filespec]
[*,PRIORITY=*base-priority]
***[*,WORKSET=*pages]*)**

Includes SYSGEN ACP and detached process parameters the system manager can set or modify. The SYSGEN parameters that affect ACPs are dynamic. The Client applies the ACP parameters only at the initial start of an ACP and ignores them in subsequent mount requests when the Client uses the same ACP.

/ADF=*option*
/NOADF

Controls whether you want to use attributes data files (ADFs). These files appear on a non-VMS server as `.$ADF$`*`filename`* files and the server uses them to store OpenVMS file attributes. You cannot directly view these files on the client system. The possible ADF *option* values are:

CREATE (the default and forced if /SERVER_TYPE=TCPWARE)	If ADFs exist on the server, the Client will use, update, and create them for new files.
UPDATE	If ADFs exist on the server, the Client will use and update them, but not create them for new files.
USE	If ADFs exist on the server, the Client will use them, but not update them nor create them for new files.

Avoid using UPDATE and USE. The Client may create ADFs anyway in certain cases, such as when renaming files. Also, changing VMS attributes for a hard-linked file may result in inconsistent VMS attributes between the linked files.

/AUTOMOUNT*[*=(INACTIVITY=*inactive-time*)*]*

Mounts a server filesystem automatically and transparently when you obtain the pathname. INACTIVITY specifies a maximum inactive period for the mount attempt. When the Client reaches this period, it unmounts the pathname. Specify the time in delta (see Table 12-12). The default is five minutes (:5). Seconds are rounded to the nearest minute.

/BACKGROUND*[*=(DELAY=*delay-time*,RETRY=*retries*)*]*

Attempts to mount the filesystem at least once in background mode. If the first mount attempt fails, it informs you and keeps retrying after an optionally specified time delay and number of retries. If omitted, the DELAY defaults to 30 seconds (::30 in delta time). The maximum delay period you can specify is approximately 49 days. The default RETRY times value is 10. If you specify RETRY=0, the Client uses 1 instead.

/CACHE_TIMEOUT*[*=(*[*DIRECTORY=*t][*,ATTRIBUTE=*t][*,READ_DIRECTORY*])*)*]*

Caching timeout information for the mount point. The following keywords apply:

The DIRECTORY timer	Specifies the amount of time (*t*) the Client waits between rereading a directory's status or contents. Specify the time in delta format (see Table 12-12). The default is 30 seconds (`::30` in delta time).
The ATTRIBUTE timer	Specifies the amount of delta time (*t*) the Client waits between rereading a file's attributes from the server. The default is 15 seconds (::15 in delta time)
The READ_DIRECTORY keyword	Forces the Client to read the contents of the directory requested when the cache timeout occurs, rather than relying on the directory's modified time. By reading the directory contents, the client can be aware of any changes to the number of files within the directory even if the directory's modify time was not updated.

/CONVERT={ STREAM_LF (default) **| STREAM_CRLF }**
/NOCONVERT (forced for TCPware's NFS Server)

Controls whether the Client should convert sequential, variable-length, carriage return carriage control (VAR-CR) files to STREAM-LF files for UNIX system servers or STREAM_CRLF for PC system servers. Some OpenVMS applications require that certain files remain VAR-CR. The default is /CONVERT=STREAM_LF unless you use /SERVER_TYPE=TCPWARE, in which case TCPware forces a /NOCONVERT.

You can only convert files opened using RMS sequential access to STREAM-LF or STREAM_CRLF format when written by the client.

The NFS Client does not perform conversions when "block mode transfers" are performed. COPY and EDT use block mode transfers when copying or creating files. Instead of COPY, use the CONVERT command. Instead of EDT, use the TPU command. Most applications do RMS sequential access when they create files on the export and these will be converted.

/DATA=*[(]read-bytes[,write-bytes][)]*

Largest amount of NFS data received (*read-bytes*) or transmitted (*write-bytes*) in a single network operation. The default for both is 8192 bytes, the maximum allowable value appropriate for most servers. The minimum is 512. If you specify only one value, that value applies to both *read* and *write*. However, you can use different values for each.

You do not normally need to use the /DATA qualifier unless a remote server imposes a restriction on data size. Also, if the NFS server requests a smaller transfer size than the one set with this qualifier, the server's requested value will override the one set by /DATA.

/FILEIDS={UNIQUE (default) | NONUNIQUE}

With UNIQUE (the default), the client uses filenames and 32-bit NFS file IDs when processing the directory information returned by the server, to determine whether cached information is valid.

With NONUNIQUE, the client uses file handles instead of file IDs in retrieving directory information. This can refresh directory entries in the client's cache more quickly, resulting in fewer "no such file" errors. However, this can degrade performance since the Client must issue additional RPC requests. /FILEIDS=NONUNIQUE automatically implies a /LOOKUPS, so do not use it together with an explicit /NOLOOKUPS.

/FORCE
/NOFORCE (default)

Controls whether or not to force an overmount or a mount that can cause filesystem occlusion. This qualifier requires OPER privilege. Overmounting a /SYSTEM mount requires SYSNAM privilege. Overmounting a /GROUP mount requires GRPNAM privilege.

/GID=*gid*

Default GID if no GID mapping exists for file access. The default value is **`-2`**. Requires OPER privileges.

/GROUP

Places the logical name in the group logical name table. If the mount is the first group or system mount on the volume, /GROUP marks the volume as group-mounted and increments the mount count. Requires GRPNAM privilege. Do not use with /SYSTEM.

/LABEL=*volume-label*

ODS-2 volume label used for the remote pathname. You can use this qualifier to provide a unique volume label on a system where there is a conflict. The default is the first 12 characters of the combined *server*:*mountpoint* parameter. The Client accepts only the first 12 characters for all other entries. The Client applies the /LABEL qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device.

/LOCK
/NOLOCK (default)

Specifies whether the Client should use advisory network file locking by way of the Network Lock Manager (NLM) to coordinate access to server files.

/NOLOOKUPS (default)
/LOOKUPS

With /NOLOOKUPS (the default), the Client does not look up file handles when building directory caches. However, when accessing an individual file, it does look up its file handle; and with a directory operation, it still looks up the handle for every file in the directory. Do not use an explicit /NOLOOKUPS together with /FILEIDS=NONUNIQUE.

/OWNER_UIC=*uic*

Specifies the UIC assigned ownership of the volume while you mount it, thereby overriding the ownership recorded on the volume. The Client applies the /OWNER_UIC qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device.

/PROCESSOR={UNIQUE | SAME:*nfs-device* | FILE:*filespec*}

Requests that NFSMOUNT associate an Ancillary Control Process (ACP) to process the volume, which overrides the default manner in which the Client associates ACPs with NFS devices. The qualifier requires OPER privilege. The possible keyword values are:

UNIQUE	Creates a new ACP (additional address space) for the new NFS device. This is useful for mounting large remote filesystems so that you can accommodate more cached information. (See *Cache Space*.)
SAME:*nfs-device*	Uses the same ACP as the specified device. The *nfs-device* specified cannot be mounted as UNIQUE.
FILE:*filespec*	Creates a new ACP running the image specified by a particular file. You cannot use wildcards, node names, and directory names in the *filespec*. Requires CMKRNL or OPER privilege.

/PROTECTION=*protection-code*

Protection code assigned the volume, following the standard syntax rules for specifying protection. If you omit a protection category, the Client denies that category of user access. The default is **`(S:RWED,O:RWED,G:RWED,W:RWED)`**.

The Client applies the /PROTECTION qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device. /PROTECTION requires OPER privilege.

/RETRIES=*max-retries*

Maximum number of times the Client retransmits an RPC request. The default is zero (0), where the Client retries the request indefinitely.

/SERVER_TYPE=*server-type*

Type of server from which the Client mounts data. The valid values for *server-type* are:

UNIX	TCPWARE	IBM_VM

The default is either UNIX or TCPWARE (if the server runs TCPware's Server).

With /SERVER_TYPE=TCPWARE, TCPware forces /NOCONVERT and /ADF=CREATE regardless of their specified settings.

/SHARE

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, causing the volume mount count to be decremented. (See *Shared*.)

/SUPERUSER=*uid*
/NOSUPERUSER (default)

Controls whether the Client maps users with SYSPRV, BYPASS, or READALL privileges to the superuser UID. The server must allow superuser access. The normal superuser UID is 0.

/SYSTEM (default)

Places the logical name in the system logical name table (the default action). If the mount is the first group or system mount on the volume, this marks the volume as system mounted and increments the volume mount count. Requires SYSNAM privilege. Do no use with /GROUP.

/TIMEOUT=*timeout-period*

Minimum timeout period (in OpenVMS delta time) for initial RPC request retransmissions. The default is ::1 (one second).

The *timeout-period* value should reflect the estimated typical round trip time for RPC requests. For slower speed links (like NFS traffic over SLIP or WANs), a larger value than the default would be appropriate.

For example, for a maximum read/write size of 8192 (see the /DATA qualifier) over a 19,200-baud SLIP line, the absolute minimum timeout value should be:

```
10240 bytes  * 8 bits per byte
--------------------------------------  =  4.27 seconds
19200 bits per second
```

The 10240 bytes are 8192 data bytes plus the worst case RPC overhead of 1048 bytes. Since 4.27 seconds is the absolute minimum, a more realistic value for this link would be in the range of 15 to 30 seconds to allow for other traffic.

/TRANSPORT=*protocol-type*

Network protocol used to transfer the data. The valid values are **TCP** and **UDP** (the default).

/UID=*uid*

Default UID, if no UID mapping exists for file access. The default value is -2. Requires OPER privileges.

/USER=*username*

Existing OpenVMS account to which the Client maps unknown UIDs. The default is the USER account. If the Client does not find the USER account, the DECNET account becomes the default. If the Client does not find the DECNET account, [200,200] becomes the default.

/VERSION (default)
/NOVERSION

Use the /NOVERSION qualifier to enforce a limit of one version on a file. This is a way of imposing an NFS file versioning scheme on OpenVMS files. /VERSION, allowing multiple versions, is the default. This qualifier is disabled if connected to a TCPware NFS server. (See *Limiting File Versions*.)

/WRITE (default)
/NOWRITE

Allows that you mount the filesystem either with write access (/WRITE) or read-only (/NOWRITE) on the local machine. If /NOWRITE, file creation, deletion, and other modifications are not allowed.

Examples

1 In this example, the Client mounts the /usr filesystem from sigma onto the OpenVMS mount point when it references the pathname. The Client keeps the path mounted until the Client reaches an inactive period of 10 minutes, after which it unmounts the pathname. Subsequent references cause the Client to remount the filesystem.

```
$ NFSMOUNT SIGMA "/usr" NFS0: /AUTOMOUNT=(INACTIVITY=00:10:00)
```

2 This example shows an overmount. The second mount specifies a lower level in the server path.

```
$ NFSMOUNT SIGMA "/usr" NFS1:[USERS.MNT]
%NFSMOUNT-S-MOUNTED, /usr mounted on _NFS1:[USERS.MNT]
$ NFSMOUNT SIGMA "/usr/users" NFS1:[USERS.MNT] /FORCE
%NFSMOUNT-S-REMOUNTED, _NFS1:[USERS.MNT] remounted as /usr/users on
SIGMA
```

3 This example shows an occluded mount. The mount point specification is "backed up" one subdirectory on the second mount. Both mounts are visible in an NFSMOUNT/SHOW. However, if you do a directory listing on NFS2:[USERS.SMITH], the [MNT] directory is no longer visible. To make the directory visible again, dismount NFS2:[USERS.SMITH].

```
$ NFSMOUNT SIGMA "/usr" NFS2:[USERS.SMITH.MNT]
%NFSMOUNT-S-MOUNTED, /usr mounted on _NFS2:[USERS.SMITH.MNT]
$ NFSMOUNT SIGMA "/usr" NFS2:[USERS.SMITH] /FORCE
%NFSMOUNT-S-MOUNTED, /usr mounted on _NFS2:[USERS.SMITH]
-TCPWARE-I-OCCLUDED, previous contents of _NFS2:[USERS.SMITH]
occluded
```

NFSMOUNT /CONFIG

Mounts one or more remote NFS directories based on information in a configuration file. In this way, you can maintain a regular list of server filesystems that you can automatically mount using one command.

DCL Format

$ NFSMOUNT /CONFIG=*filespec*

Parameter

filespec

OpenVMS file containing the configuration information. The contents of the file should include line entries in the format prescribed by the NFSMOUNT command:

***server "nfs-path" mountpoint** [**logical**] [**qualifiers**]*

The configuration file must have complete information for a mount on each line (continuation lines are not allowed). The Client ignores blank or comment lines. Mount requests in the file can have further configuration file references, although there is limited nesting of these requests.

Qualifiers

Note! The Client uses qualifiers specified with the NFSMOUNT /CONFIG command as defaults for mount requests in the configuration file. However, qualifiers included with mount requests in the file override these defaults.

See the NFSMOUNT command for details on the following qualifiers:

/ACP_PARAMS=(*[*BUFFER_LIMIT=*limit-value]*

[*,DUMP*]
[*,IO_DIRECT=*value]
[*,IO_BUFFERED=*value]
[*,MAX_WORKSET=*pages]
[*,PAGE_FILE=*filespec]
[*,PRIORITY=*base-priority]
***[*,WORKSET=*pages]*)**

/ADF=*option*
/NOADF

/AUTOMOUNT*[*=(INACTIVITY=*inactive-time*)*]*

/BACKGROUND*[*=(DELAY=*delay-time*,RETRY=*retries*)*]*

/CACHE_TIMEOUT*[*=(*[*DIRECTORY=*t][*,ATTRIBUTE=*t]*)*]*

/CONVERT={ STREAM_LF | STREAM_CRLF }
/NOCONVERT

/DATA=*[(]read-bytes[,write-bytes][)]*

/FILEIDS={UNIQUE | NONUNIQUE}

/FORCE
/NOFORCE

/GID=*gid*

/GROUP

/LABEL=*volume-label*

/LOCK
/NOLOCK

/LOOKUPS
/NOLOOKUPS

/OWNER_UIC=*uic*

/PROCESSOR=*keyword*

/PROTECTION=*protection-code*

/RETRIES=*max-retries*

/SERVER_TYPE=*server-type*

/SHARE

/SUPERUSER=*uid*
/NOSUPERUSER

/SYSTEM

/TIMEOUT=*timeout-period*

/TRANSPORT=*protocol-type*

/UID=*uid*

/USER=*username*

/WRITE
/NOWRITE

Examples

1 The following command consults the CONFIG_NFS.TXT file for mounting information.

```
$ NFSMOUNT /CONFIG=CONFIG_NFS.TXT
```

2 The following command also sets data size and username parameters (which can be overridden by qualifiers in the configuration file).

```
$ NFSMOUNT /CONFIG=CONFIG_NFS.TXT /DATA=512 /USER=BART
```

NFSMOUNT /SHOW

Displays the mounted directories at all mount points or at a particular mount point.

DCL Format

$ NFSMOUNT /SHOW *[mountpoint | device:]*

Parameters

mountpoint

Full NFS device name and directory tree for which to show mount information. For example:

```
NFS1:[USER.NOTES]
```

Alternately, you can use a logical name for the mount point.

device:

NFS device name part of the *mountpoint* parameter (such as NFS1:).

Alternately, you can use a logical name for the mount point. With the /ALL qualifier, the Client uses only the device portion of the logical name.

Qualifiers

/ALL

Shows mount information for all servers, or a specified server or NFS device.

/FULL

Displays the full, current operating parameters related to each mount.

See the NFSMOUNT command for descriptions of the qualifiers that correspond to each of the operating parameters.

/QUOTA

Displays quota information for the current user's mount. The qualifier used by itself shows four columns at the top of the display indicating the block usage, soft limit (quota), hard limit, and grace period.

Use /QUOTA with the /FULL qualifier to show four additional columns indicating any possible file quotas. These show as zeros for an OpenVMS system but as actual values for UNIX systems that support file quotas.

Use /QUOTA with the /USER qualifier to request quotas for other than the default user.

/USER=*username*

Use with /QUOTA to show quotas for a specific user. This requires the mount to have been performed using the /SUPERVISOR qualifier, which maps users with SYSPRV, BYPASS, or READALL privileges to the superuser UID. /USER requires SYSPRV or GRPPRV privileges.

Examples

1 This example provides the default command display.

```
$ NFSMOUNT /SHOW
_NFS1:[000000]  automount (inactivity timer  0 00:23:00.00), mounted
SIGMA.PROCESS.COM:/usr
_NFS2:[000000]  mounted
IRIS.PROCESS.COM:/usr/users
```

2 This example shows characteristics of all mounts on a specific NFS device.

```
$ NFSMOUNT /SHOW NFS0: /ALL
_NFS1:[A.B]  mounted
SIGMA.PROCESS.COM:/usr
_NFS2:[A.C]  mounted
SIGMA.PROCESS.COM:/work
```

3 This example shows the full mount display with all operating parameters for a specific NFS device. Note that you can either enable or disable `Writing` and `Write conversion`.

```
$ NFSMOUNT /SHOW NFS1: /FULL
_NFS1:[000000]  mounted
  MERAK.PROCESS.COM:/eng/nfsuser
  Transport                  UDP   Writing              Enabled
  Read/write size      8192/8192   Write conversion    Disabled
  RPC timeout      0 00:00:01.00   ADF usage  USE,UPDATE,CREATE
  RPC retry limit              0   Fileids    Unique, Nolookups
  Attribute time   0 00:00:15.00   Server type          TCPware
  Directory time   0 00:00:30.00   Advisory Locking    Disabled
  Cache Validation   MODIFY TIME   Default user          [USER]
  Superuser                   No   Default UID,GID       100,15
```

4 This example shows the additional full block and file quotas for the user’s mount.

```
$ NFSMOUNT /SHOW NFS2: /QUOTA /FULL
_NFS2:[000000]  mounted
  viola:/pctest
  Disk Quotas for user [SMITH]: (inactive)
  Blocks   Quota   Limit   Grace   Files   Quota   Limit   Grace
  117355   500000  600000          0       0       0
  Transport                  UDP   Writing               Enabled
  Read/write size      8192/8192   Write conversion     Disabled
  RPC timeout      0 00:00:01.00   ADF usage   USE,UPDATE,CREATE
  RPC retry limit              0   Fileids     Unique, Nolookups
  Attribute time   0 00:00:15.00   Server type           TCPware
  Directory time   0 00:00:30.00   Advisory Locking     Disabled
  Cache Validation   MODIFY TIME   Default user           [USER]
  Superuser                   No   Default UID,GID        100,15
```

NFSDISMOUNT

Dismounts an NFS mount point from the local device and directory structure.

DCL Format

$ NFSDISMOUNT *[mountpoint* | *device*:*]*

Parameters

mountpoint

Full NFS device name and directory tree to dismount. For example:

```
NFS1:[USER.NOTES]
```

You can also use a logical name for the mount point. At the end of the NFSDISMOUNT operation, the Client deletes the logical name from the job logical name table.

device:

NFS device name part of the *mountpoint* parameter (such as NFS1:). You can use the *device*: alone only with the /ALL qualifier.

Alternately, you can use a logical name for the device specification. TCPware considers only the NFS device part of the logical name.

Qualifiers

/ALL

Dismounts all filesystems from all servers, or a specified server or NFS device. The following options are available:

NFSDISMOUNT /ALL	Dismounts all filesystems from all servers
NFSDISMOUNT /ALL /HOST=*server*	Dismounts all filesystems on the specified server. (See the /HOST qualifier below.)
NFSDISMOUNT *device*: /ALL	Dismounts all filesystems on the specified device (such as NFS1:).

Note! Dismounting a /SYSTEM mount requires SYSNAM privilege. Dismounting a /GROUP mount requires GRPNAM privilege.

/HOST=*server*

When used with the /ALL qualifier, dismounts all filesystems from the specified server. The /HOST qualifier is otherwise meaningless.

/WAIT
/NOWAIT (default)

Specifies whether or not to dismount the mounted filesystem if there are still outstanding activities.

With /WAIT, the command waits until the Client completes the dismount. If you try to open any files on the mount point, the dismount fails.

With /NOWAIT, the Client completes the command immediately. However, the dismount does not actually occur until all file activity has completed.

Examples

1. This example dismounts the specified mount point only. The Client dismounts only [USR.MNT] and no other mount in the directory structure of the NFS3: device.

```
$ NFSDISMOUNT NFS3:[USR.MNT]
```

2. This example dismounts the NFS1:[000000] mount point and waits for it to occur.

```
$ NFSDISMOUNT NFS1: /WAIT
```

3. This example dismounts all mount points on all devices.

```
$ NFSDISMOUNT /ALL
```

4. This example dismounts all mount points served by host SIGMA.

```
$ NFSDISMOUNT /ALL /HOST=SIGMA
```

Chapter 13

Managing NFS-OpenVMS Server

Introduction

This chapter describes how to manage the NFS-OpenVMS Server. It includes the following topics:

- Server security
- Mounting directories on a client
- Network file locking
- Managing Server parameters
- Maintaining databases
- PCNFSD services and remote printing
- Filename mapping
- Server implementation of NFS protocols

See the *NETCU Command Reference*, Table 1-4, for a description of the commands used to manage NFS servers and clients.

Server Security

The Server provides several features that maintain the integrity of the OpenVMS filesystem.

First, the Server requires that the local system must register any user trying to access OpenVMS files. You do this through the PROXY database when you configure the Server and through later modifications as needed.

Second, you must export an OpenVMS directory for an NFS user to access it. The Server does this through the EXPORT database when you configure the Server and through later modifications as needed.

You can take the following additional system security measures:

- Assign an NFS rights identifier to further restrict file access (see the NFS_ACCESS_IDENTIFIER logical under *Server Parameters*).

- Require all Remote Procedure Call (RPC) requests to originate from privileged ports (see Table 13-6).
- Restrict all remote mounts to the NFS superuser only (see Table 13-6).
- Restrict mounts only to explicit directories and not their subdirectories (see Table 13-6).
- Require the PROXY database to define the mount requester's identification (see the next section).

PROXY Database

The PROXY database maps OpenVMS user identification to NFS user identification. NFS user identification is different from that of OpenVMS in that it follows the UNIX model.

OpenVMS identifies users by a username and user identification code (UIC). The UIC consists of a group and a member number. An OpenVMS user can belong to only one group, which can have many members.

NFS follows the UNIX model in identifying users by user ID (UID) and group ID (GID) numbers. An NFS user can belong to many groups, and thus have several GIDs. Each NFS request includes the NFS user's effective UID, GID, or list of GIDs. You can find users' UIDs and GIDs in the `/etc/passwd` file on the UNIX client.

The Server uses the PROXY database:

When an NFS user requests access to the OpenVMS filesystem	TCPware maps an NFS user's UID and GID to an OpenVMS username and UIC. The Server uses the UIC to check file access permission.
When the NFS client requests file attributes from the server	The Server maps the file owner's UIC to a UID/GID pair.
When a PC requests authentication using PCNFSD	The Server uses the username and password to validate the user, and the UIC to find and return a UID/GID pair.

Maintaining PROXY

The Server creates an empty PROXY database during installation. You maintain the PROXY database with the ADD PROXY, CREATE PROXY, REMOVE PROXY, and SHOW PROXY commands in TCPware Network Control Utility (NETCU).

A PROXY database entry specifies an OpenVMS username and a corresponding NFS user's UID and GID. The /HOST qualifier with the ADD PROXY command also lets you specify the name of the hosts or hosts where the user account is valid.

The following example shows how to use the ADD PROXY command to assign the SMITH OpenVMS account to an NFS user with a UID=210 and GID=15 on host `tulip`:

```
$ NETCU
NETCU> ADD PROXY SMITH /UID=210 /GID=15 /HOST=TULIP
```

The PROXY database must contain an entry for each NFS user, including the superuser (see the next subsection).

When you add entries to the PROXY database:

- The OpenVMS username determines file access rights, not the NFS user's UID and GID. The NFS user account has the same access rights as are assigned the OpenVMS account.
- Assign each NFS user the same UID/GID on each NFS client. (See your NFS client documentation for details on global user ID space.)
- Avoid using wildcard UIDs or GIDs. A one-to-one mapping between OpenVMS users and NFS users is easier to maintain.
- Use the /HOST qualifier to allow access only to users from a particular host.
- For PCNFSD users, assign an arbitrary UID and GID for each PC user. Choose a unique UID for each user. Give the same GID to users that need to have group access to each other's files.

Adding Superusers

The superuser (or `root`) is a UNIX system user with UID=0 who can perform any operation on a file or process on the client system. However, the superuser cannot automatically access the OpenVMS filesystem on the server. The PROXY database must register a superuser.

The NFS convention is to replace the superuser's UID/GID pair (UID=0, and any GID) with the default values of UID=-2 and GID=-2. By UNIX conventions, this translates to user `nobody`, which gives the superuser limited access rights. To register a superuser, you must use UID=0 and GID=1, as follows:

```
NETCU> ADD PROXY DECNET /UID=0 /GID=1
```

The OpenVMS account to which you assign the superuser access rights determines what rights a superuser has on the OpenVMS system. Superusers require enough access rights so that they can mount directories. (In fact, some server configurations restrict mounting to the superuser). Also, when a user runs a `setuid` to root program, the UID/GID in any resulting NFS request has the root UID and, therefore, requires superuser access.

You can create a PROXY entry for a superuser that provides limited access to an OpenVMS filesystem but still allows a superuser to mount exported directories. One example is the DECNET account. Alternately, you can use the OpenVMS AUTHORIZE command to add an account for the superuser on the OpenVMS host.

If you have trusted superusers at particular hosts and wish to give them full privileges on the OpenVMS system, add a separate superuser entry. Assign the superuser to a privileged account (such as SYSTEM) and use the /HOST qualifier to restrict access to a specified host. In the following example, only the superuser on `lilac` has SYSTEM account privileges:

```
NETCU> ADD PROXY SYSTEM /UID=0 /GID=1 /HOST=LILAC
```

Reloading PROXY

The PROXY database is normally static. This means that you have to reload the database every time you use ADD PROXY or REMOVE PROXY to change it. However, you can opt to update the

PROXY database dynamically (make it dynamic). You can do so in two ways:

1	Define the TCPWARE_NFS_DYNAMIC_PROXY logical to enable dynamic PROXY database reloading, as follows: `$ DEFINE/SYSTEM/EXEC TCPWARE_NFS_DYNAMIC_PROXY keyword[,keyword]` The *keywords* are CLIENT, SERVER, NOCLIENT, and NOSERVER, used in any reasonable combination. Use CLIENT to enable Client reloading and SERVER to enable Server reloading. However, the /NOCLIENT and /NOSERVER qualifiers used with the ADD PROXY or REMOVE PROXY commands override the logical setting.
2	Use the /CLIENT or /SERVER qualifiers with the ADD PROXY or REMOVE PROXY commands. You can also mix and match by using /CLIENT with /NOSERVER, /NOCLIENT with /SERVER, and so on. Here is an example of its use: `$ NETCU ADD PROXY SMITH /UID=210 /GID=5 /NOCLIENT /NOSERVER` If you disable PROXY database reloading on either the Client or Server, both of these methods requires the RELOAD PROXY command. RELOAD PROXY is best used if you also specify a username parameter, so that you can reload for a specific username only. Otherwise, it reloads the entire database into memory each time. Therefore, it is best to use RELOAD PROXY at the initial configuration, and only sparingly thereafter.

EXPORT Database

The EXPORT database contains entries that specify an OpenVMS directory and the host or group of hosts allowed to mount that directory. More than one host can access a directory. The EXPORT database differs from the PROXY database in that the Server grants access to a host rather than to a user. If an OpenVMS directory is not in the EXPORT database, an NFS client cannot mount that directory.

An EXPORT database entry specifies a pathname for the OpenVMS directory. Because the OpenVMS device and directory specifications differ from those NFS clients use, the Server lets you reference the OpenVMS directory by a UNIX-style pathname. You can assign any pathname to the OpenVMS directory.

CAUTION! An authorized user at a remote host can access all subdirectories and files below the export point you specify. Unless you work in a trusted environment, do not export a top level directory, even though it may seem easier to do so. Export only the level of directories that the remote users need, and none higher.

Maintaining EXPORT

The Server creates an empty EXPORT database during installation. You maintain the EXPORT database using the ADD EXPORT, CREATE EXPORT, REMOVE EXPORT, RELOAD

EXPORT, and SHOW EXPORT commands in NETCU. For example, the following command places an entry in the EXPORT database:

```
NETCU> ADD EXPORT "/work/notes" $DISK2:[WORK.NOTES] -
_NETCU> /HOST=(ORCHID, ROSE)
```

This command exports the OpenVMS directory $DISK2:[WORK.NOTES] as path `"/work/notes"` to hosts ORCHID and ROSE. The pathname is an arbitrary one selected to reference the OpenVMS directory. The ADD EXPORT command requires that you enclose the pathname in quotes.

When a client mounts a subdirectory of an exported directory, each element in the path beyond the exported path must match the corresponding OpenVMS subdirectory name. Separate each element with a slash (/). For example, suppose the NFS client mounts:

```
$ DISK2:[WORK.NOTES.LETTERS.STUFF]
```

To match `/work/notes`, the NFS client uses this path:

```
/work/notes/letters/stuff
```

The NFS filename mapping rules apply to the path elements below the export point.

Reloading EXPORT

Updating the EXPORT database (using ADD EXPORT or REMOVE EXPORT) usually updates only the server on the host executing the command. You must use either the RELOAD EXPORT command, or restart all the other servers on the cluster to implement changes to the EXPORT database on them.

However, you can automatically reload updates to the shared database on the cluster by setting the TCPWARE_NFS_DYNAMIC_EXPORT logical to CLUSTER, as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_NFS_DYNAMIC_EXPORT CLUSTER
```

This causes the Server to use locks to communicate changes to all the servers on the cluster. The default for TCPWARE_NFS_DYNAMIC_EXPORT is LOCAL (not to use locks).

EXPORT Options

The options you can specify while adding entries to the EXPORT database are as follows, using the indicated ADD EXPORT command qualifiers:

- If you want only specified host or hosts to access the exported OpenVMS directory, use the /HOST=*[host[,host,...]* qualifier.
- Whether or not to enable on-the-fly file conversion: /*[*NO*]*CONVERT.
- Whether or not clients can mount subdirectories of a mount point: /*[*NO*]*EXPLICIT_MOUNT.
- What kind of filename mapping you want to use: /FILENAME=*option*.

  The NFS-OpenVMS Server includes the UPPERCASE keyword for this qualifier.

UPPERCASE changes the default case for exported filenames from lowercase to uppercase, for SRI filename mappings only. The full syntax of the command is

```
$ NETCU ADD EXPORT /FILENAME=(SRI, UPPERCASE)
```

Examples of filename conversions are as follows:

VMS Name	Lowercase	Uppercase
foobar.txt	foobar.txt	FOOBAR.TXT
$foobar.txt	FOOBAR.TXT	foobar.txt
foo$bar.txt	fooBAR.TXT	FOObar.txt

- Whether or not you want only the highest version files to appear in a directory request: /*[*NO*]*HIGHEST_VERSION.
- Whether or not you want incoming requests to originate from a privileged port: /*[*NO*]*PRIVILEGED_PORT.
- Whether or not you want mount requests to originate from a user mapped in the PROXY database: /*[*NO*]*PROXY_CHECK.
- What kind of record format you want to use for newly created files: /RFM=*options*.
- Whether or not you want the server (and not just the client) to perform file access checking: /*[*NO*]*SERVER_ACCESS.
- Whether or not you want only the superuser to mount filesystems: /*[*NO*]*SUPERUSER_MOUNT.
- Whether or not the filesystem should be read-only: /*[*NO*]*WRITE.

PCNFSD Services

This section describes PCNFSD authentication services and how to configure for remote printing services. The PCNFSD server supports both the PCNFSD Version 1 and Version 2 protocols. (Version 2 offers enhanced printing features.)

PCs and other NFS clients use PCNFSD if they do not have multiuser accounts or do not provide user authentication. PCNFSD lets the client user obtain the UID/GID that the NFS protocol requires. It also provides remote print spooling services.

The `Do you want PCNFSD enabled?` prompt in the Server configuration procedure allows a YES, NO, and PRINTING-ONLY response. PRINTING-ONLY enables print spooling of files on the server without enabling PCNFSD authentication.

If you configure PRINTING-ONLY, PCNFSD simply discards `auth` requests. Use this support primarily when you do not want the Server to respond to PCNFS `auth` requests sent to a broadcast address.

PCNFSD Authentication

To use the Server, the PC user must obtain a UID and GID through PCNFSD authentication. The PC user provides a valid OpenVMS username and password, and PCNFSD provides the UID and GID.

See your PC documentation for the command to specify the username and password.

PCNFSD checks the OpenVMS User Authorization File (UAF) to validate the username and password. If these are valid, PCNFSD uses the UIC to return the corresponding UID/GID from the PROXY database.

When you create an entry for a PC user in the PROXY database, assign any UID. Assign a unique UID to each user and give the same GID to users that need to have group access to files. The UID and GID cannot be wildcards.

If PCNFSD cannot validate a user for any reason, it writes an error message to the Server log file. This message includes the username and internet address of the remote host that issued the request.

Remote PC Printing

PC users can use PCNFSD for remote printing if you:

- Create a spool directory to hold the files you want printed, as well as a subdirectory for each PC client.
- Make sure that this spool directory (or a subdirectory) is in the EXPORT database. Note that the EXPORT database should not list the subdirectory for each client host.

  For printing large application files from the PC, we recommend adding the EXPORT option using the following NETCU command:

  ```
  ADD EXPORT "/spool" device:[directory] /NOCONVERT /RFM=UNDEFINED
  ```

  This overcomes an OpenVMS file conversion buffer error that may occur, especially with files over 32,768 bytes, which the SYSGEN parameter PQL_MBYTLM reflects.
- Define the NFS_PCNFSD_SPOOL parameter, either during configuration at the `Enter the spool directory:` prompt or by defining the TCPWARE_PCNFSD_SPOOL logical.

  The parameter value must match the NFS pathname to the newly created spool directory. This pathname must be mapped to the OpenVMS spool directory in the EXPORT database. Make sure you enable the NFS_PCNFSD_ENABLE parameter before defining NFS_PCNFSD_SPOOL.

See your PC documentation for printing information for your particular PC.

Mounting Client Directories

NFS clients access OpenVMS files on the NFS server by mounting directories on the client. The MOUNT protocol services the mount request.

Mounting procedures vary by client and may require superuser privileges, or in the case of PC clients, a username and password. Some clients mount a remote directory automatically when they reboot the system (as in the case of `fstab`). Others mount a remote directory dynamically when they reference the remote file (as with an automount).

Mount procedures require the following information:

- The pathname of the exported directory that matches the pathname in the EXPORT database
- The name of the host running the server that contains the files you want mounted
- A pathname on the client designated as the mount point

Example 13-1 shows a mount command provided by TCPware NFS-OpenVMS Client:

Example 13-1 NFS-OpenVMS Client Mount Command

```
NFSMOUNT IRIS "/WORK/RECORDS" NFS0:[USERS.MNT]
```

In the example, IRIS is the name of the OpenVMS server host. `/WORK/RECORDS` is the pathname of the exported directory. NFS0:[USERS.MNT] is the mount point on the OpenVMS client host.

Check your NFS client documentation before mounting directories. Mount commands and procedures vary by operating system. Chapter 12, *NFS-OpenVMS Client Management*, *Client Commands* describes the client mount commands.

Network File Locking

The Server supports file locking through its implementation of the Network Lock Manager (NLM) and Network Status Monitor (NSM) protocols. Many NFS client systems support file locking, even on the record and byte level, as long as the byte ranges do not overlap. File locking on the Server is multi-threaded, where the Server can satisfy more than one lock request at a time.

NFS file locking is only advisory. When a client requests a lock on a server file, the goal is for one of its processes to gain exclusive access to this file (or part of the file) and force other processes to wait until the original process releases the lock again. However, the only way NFS denies a client user access to a locked file is if the user also requests a lock on it.

There are two views on network file locking, one from the NFS client's viewpoint and one from the OpenVMS resident user's viewpoint. (See the following sections.)

NFS Client Users' View

When an NFS client user requests an advisory lock on a server filesystem, this sends a `lockd` request to the NLM of the server also running NFSD. This server checks its lock database to see if it can grant the lock. The server cannot grant the lock if:

- Another client has the same file (or region or byte range of the file) already locked.
- An OpenVMS user has the same file open for exclusive access.
- The server waits to reclaim locks during the grace period described below.

The Server also includes a Network Status Monitor (NSM). The NSM cooperates with other status monitors on the network to notify the NLM of any changes in system status (such as when a crash occurs).

For example, if the server crashes and comes back up, the server NSM notifies the client NSM that it should resend requests for locks in place before the crash, within a certain grace period (usually 45 seconds). You can request new locks only after this grace period. However, if a client with mounted server files crashes, nobody knows to resend lock requests until the client comes back up again.

OpenVMS Users' View

To prevent OpenVMS users from accessing files that NFS clients have locked, the Server's NLM requests NFSD to open these files for exclusive access. This essentially prevents all access to these files by OpenVMS users. When the client releases the lock by closing the file, the NLM requests NFSD to close the file, at which point OpenVMS users again access it.

If network file locking is to occur in a VMScluster environment, we advise exporting a filesystem from a single node in the cluster only. This way, only a single OpenVMS exclusive lock need occur. Client users can then apply locks on files (or parts of files if enabled) without conflicting with exclusive locks applied from other nodes.

Mapping Filenames

Once you mount a filesystem, the Server tries to make the client files recognizable in OpenVMS. Often the filename syntax for NFS files is very different from that of OpenVMS files. For example, NFS filenames do not include file version numbers.

The Server translates (maps) filenames from the client so that your OpenVMS host can recognize and use them. Three types of mapping schemes are available:

- Stanford Research Institute (SRI) International mapping, the default scheme between NFS and OpenVMS systems
- PATHWORKS case-insensitive mapping (PATHWORKS)
- PATHWORKS case-sensitive mapping (PATHWORKS_CASE)

Set up the appropriate filename mapping scheme using the /FILENAME qualifier of the ADD EXPORT command in NETCU. If you do not specify the scheme using this qualifier, the Server uses the SRI International scheme by default.

Table 13-1 shows examples of how the Server maps NFS directory names and filenames using the SRI International mapping scheme. All the client files in the table are NFS files.

For the filename mapping rules, see Appendix A, *NFS-to-OpenVMS Filename Mapping*.

The filename mapping schemes for the Server and the NFS-OpenVMS Client are identical and totally compatible.

Table 13-1 Server Filename Mapping

Filename on server...	Is mapped to filename on client...
SERVERFILE.;1	serverfile
CASESHIFTEDFILE.;1	CaseShiftedFile
DOT.FILE$5NTEXT;1	dot.file.text
DOT$5NDIRECTORY$5NLIST.DIR;1	`dot.directory.list` (**identified as a directory in the UNIX listing**)
SPECIAL$5CCHAR$5FFILE.;1	special#char&file
DOLLAR$$$S$IGN$$5CFILE.;1	dollar$Sign$5cfile

Protecting Files

The Server protects an OpenVMS file by comparing its protection information with the user's identification and access rights. It then grants or denies access based on the results of these comparisons.

When an NFS user requests access to an OpenVMS file, the Server uses the PROXY database to map the user's user and group identification (UID/GID) on the remote host to a username and UIC on the OpenVMS host. In most cases, this allows the NFS user to have the same access to files as the proxy OpenVMS user.

The NFS client can also do local access checking based on its user and file information and access checking rules before sending the request to the Server host. In some cases, this results in the NFS user not having the same access to files as the proxy OpenVMS user.

The following sections explain how the Server resolves differences between the two filesystems to provide the best possible mapping between client and server.

UIC Protection

The type of access an OpenVMS user has to a file depends how the file and user UICs are related.

OpenVMS has four file ownership categories: SYSTEM, OWNER, GROUP, and WORLD. Each category can have up to four access types: read (R), write (W), execute (E), and delete (D). Each file has a protection mask that defines:

- The categories assigned to the file
- The types of access granted to each category

Here is an example of an OpenVMS protection mask:

```
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=<NO ACCESS>
```

UID/GID Protection

NFS uses a similar protection scheme as OpenVMS.

Each NFS user has a UID and GID. The file protection categories are: OWNER, GROUP, and OTHER, with the file access types of read (r), write (w) and execute (x). The NFS user's access to a file depends on how the file and owner UIDs/GIDs are related.

The Server maps OpenVMS and NFS system protection masks and user identifications so that the relationship between a user and a file remains consistent. For example, if an OpenVMS user owns a particular file and an NFS user is mapped to the account through the PROXY database, the NFS client also considers the local user to be the owner of the file.

Note! In OpenVMS, the owner of the file has absolute control over it. This also applies to files remote users create in the mounted filesystem.

OpenVMS-to-NFS File Attribute Mapping

When the NFS client requests the attributes of a file, the Server maps:

- The protection mask to an NFS protection mask
- The owner UIC to a UID/GID

Table 13-2 shows how the Server maps the protection mask from OpenVMS to NFS.

Table 13-2 OpenVMS-to-NFS Protection Mapping

OpenVMS category...	In NFS is...	With OpenVMS type...	In NFS is...
SYSTEM	(not mapped)		
OWNER	user	R	r
		W	w
		E	x
		D	(not mapped)
GROUP	group	R	r
		W	w
		E	x
		D	(not mapped)
WORLD	other	R	r
		W	w

Table 13-2 OpenVMS-to-NFS Protection Mapping (Continued)

		E	x
		D	(not mapped)

The Server does not map the OpenVMS SYSTEM category and delete (D) access type because they do not exist in the NFS system environment.

The Server maps OpenVMS execute (E) to NFS execute (x). However, the OpenVMS system uses the E access type more often than does NFS. Thus, some files might appear to be executable to an NFS host when they are not.

Table 13-3 shows the rules the Server follows to ensure that it correctly maps the UIC to the UID/GID.

If the Server cannot find the UIC in the PROXY database, or the UID or GID are wildcards, the Server returns the default UID or GID.

Table 13-3 UIC-to-UID/GID Mapping Rules

If the file's OWNER UIC...	Then the Server Returns the...
Matches the requesting NFS user's UIC	UID/GID of the requester
Group matches the requesting NFS user's UIC group	GID of the requester and returns the UID from the PROXY database
Does not match the requesting NFS user's UIC	UID/GID from the PROXY database

NFS-to-OpenVMS File Attribute Mapping

When the NFS client sets or changes the attributes of a file, the Server maps the NFS file protection mask to an OpenVMS file protection mask.

Table 13-4 shows how the Server maps the protection mask from NFS to OpenVMS.

Table 13-4 NFS-to-OpenVMS Protection Mapping

NFS category...	In OpenVMS is...	With NFS type...	In OpenVMS is...
user	OWNER/SYSTEM	r	R
		w	W
		x	E
			D (unless ADF denies) [1]
group	GROUP	r	R
		w	W
		x	E
			D (unless ADF denies) [1]
other	WORLD	r	R
		w	W
		x	E
			D (unless ADF denies) [1]

[1] The Server allows delete (D) access only if a special attributes data file (ADF) the Server may create (and associates with the file) does not explicitly deny file deletion.

Access Control Lists

Access Control List (ACL) file protection is an OpenVMS feature that grants or denies access to a file based on a rights identifier.

If a file has an ACL, the OpenVMS system first uses the ACL for protection checking. If the ACL grants or denies access, OpenVMS goes no further. If the ACL does not grant or deny access, OpenVMS checks the protection mask.

NFS clients using the OpenVMS filesystem may encounter files or directories protected by ACLs. But since the ACLs are unique to the OpenVMS system, the NFS client only checks the protection mask. If the protection mask denies access, the NFS client does not attempt access, even if the file's ACL overrides the protection.

Because the NFS client uses only the protection mask, it is recommended that OpenVMS files protected by ACLs have:

- The ACL set to deny access
- The protection mask set to allow file access

This allows the NFS client to attempt access on the basis of the protection mask, and lets the OpenVMS system control whether access is granted or denied.

When an NFS user creates a file on the OpenVMS host and the directory has an ACL that specifies +DEFAULT, the new file gets the ACL of the directory.

File Formats

The NFS protocol does not define standard file and record formats or a way of representing different types, such as text or data files. Each operating system can have a unique file structure and record format.

The Server provides access to all OpenVMS files. However, even though an NFS client can access a file, the client may not be able to correctly interpret the contents of a file because of the differences in record formats.

The UNIX operating system stores a file as a stream of bytes and uses a line feed (LF) character to mark the end of a text file line. PC systems also store a file as a stream of bytes, but use a carriage-return/line-feed (CRLF) character sequence to mark the end of a text file line. PC systems sometimes also use a Ctrl/Z character to mark the end of a file.

The OpenVMS operating system, with its Record Management Services (RMS), provides many file organizations and record formats. RMS supports sequential, relative, and indexed file organizations. It also supports FIXED, STREAM, STREAM_CR, STREAM_LF, UNDEFINED, VARIABLE, and variable with fixed size control area (VFC) files.

NFS clients most commonly need to share text files. STREAM is the RMS record format that most closely matches PC text files. STREAM_LF is the RMS record format that most closely matches UNIX text files.

In OpenVMS, you can store standard text files in VARIABLE, STREAM_LF, or VFC record format. Most OpenVMS utilities can process these text files regardless of the record format because the utilities access them through RMS.

The intent of the Server is to provide convenient access to the majority of OpenVMS files. Because many OpenVMS text files are VARIABLE or VFC format, the Server converts these files to STREAM or STREAM_LF format as it reads them.

Reading Files

The Server reads all files (except VARIABLE and VFC) block by block without interpreting or converting them. It reads VARIABLE and VFC files by converting them to STREAM or STREAM_LF, based on a selected option. The file on the NFS server remains unchanged.

The Server's automatic file conversion process can cause a slow reading of VARIABLE and VFC files. For example, in returning the file size, it reads the entire file. Full directory listings can also be slow if the directory contains a number of VARIABLE or VFC record format files. If you need frequent access to these files, consider converting them using the OpenVMS CONVERT utilities described in *Converting Files Manually*.

See the NFS_DIRREAD_LIMIT parameter in *Advanced Parameters*.

Writing Files

By default, the Server creates STREAM_LF files, but can also create STREAM files on demand. It writes all files except VARIABLE and VFC block by block without interpreting or converting them. If an NFS client tries to write to or change the size of an existing file not having STREAM, STREAM_LF, STREAM_CR, FIXED, or UNDEFINED format, the Server returns an `EINVAL` error.

Converting Files Manually

You can improve server performance by manually converting files using the OpenVMS CONVERT utilities described in this section.

Variable to STREAM_LF

Use this conversion procedure to make a variable-length file available to a UNIX system client without using the Server's automatic conversion feature. To convert a variable-length record file to STREAM_LF, the command format is:

`CONVERT/FDL=TCPWARE:STREAMLF` ***`source-file destination-file`***

The *source-file* specification is the variable-length record file. The *destination-file* specification is the name of the new file to contain the STREAM_LF records.

STREAM_LF to Variable

Use this conversion procedure to make a file created by a UNIX system client available to an OpenVMS application that does not understand the STREAM_LF record format. To convert a STREAM_LF file to variable-length, the command format is:

`CONVERT/FDL=TCPWARE:VMSTEXT` ***`source-file destination-file`***

The *source-file* specification is the STREAM_LF file. The *destination-file* specification is the name of the new file to contain the variable-length records.

Variable to STREAM

Use this conversion procedure to make an OpenVMS variable-length file available to a PC client. Keep in mind that the Server's automatic conversion procedure uses LF characters, not CRLF character sequences, for record terminators.

To convert a variable-length record file to STREAM format (with CRLF line terminators), the command format is:

```
CONVERT/FDL=TCPWARE:STREAMCRLF source-file destination-file
```

The *source-file* specification is the variable-length record file. The *destination-file* specification is the name of the new file to contain the STREAM records.

Note! The variable-to-stream conversion does not add a Ctrl/Z to the end of the file. If the PC application requires the Ctrl/Z, use the conversion program the NFS client software provides.

Server Parameters

TCPware provides several basic parameters you can adjust to better suit your needs. To change the value of any of these parameters, invoke the network configuration command procedure (CNFNET) by entering the following command:

```
$ @TCPWARE:CNFNET NFS
```

The Server also provides advanced parameters that you rarely need to change but appear here for reference purposes only.

The default parameter values appear in parentheses following the parameter name. All parameters are logicals and are static. When you make a change to a parameter, you must stop and restart the Server for the change to take effect. TCPware uses logical names (the parameter names prefixed by TCPWARE_) to communicate the parameters to the NFS server. The STARTNET procedure defines these logicals.

Basic Parameters

The basic parameters described here are in the same order in which the Server prompts you to provide values for them during the NFS configuration procedure. The default setting for each parameter appears in parentheses.

NFS_ACCESS_IDENTIFIER (`null`)	Specifies the name of a rights identifier you want assigned to all NFS users. You can then modify the access control lists (ACLs) of files to grant or deny access to holders of the rights identifier. The default is null (no rights identifier). OpenVMS files protected by ACLs should have the UIC-based protection mask set to allow file access and the ACL set to deny access. This lets the NFS client access on the basis of the protection mask, and lets the OpenVMS system control whether to grant or deny access.
NFS_SECURITY (`0`)	Enables various security features. This parameter is a bit mask value (in decimal) as defined in Table 13-5. The following global parameters supersede the values set using the corresponding qualifiers of the ADD EXPORT command, if applicable, as indicated in Table 13-5. ***CAUTION!*** Do not use bits 0 and 1 for PC clients using PCNFS. If you use PC-NFS printing with mask value=2, add an entry to the EXPORT database for each client subdirectory (not just a single entry for the spool directory.) The pathname listed in the EXPORT database should be the NFS_PCNFSD_SPOOL parameter value concatenated with the name of the client subdirectory. If you set bit 5, PC-NFS users can print to batch queues. This may present a security risk, since users could submit batch jobs under a privileged (or another) user by forcing the UID/GID values of their choice. Disabling use of the intrusion database for PCNFSD, by setting bit 6, affects all exports. A bit mask 8 value of 128 disables PCNFSD deletion of printed files from the spool directory.
NFS_LOG_CLASS (`-1`)	Enables the type of information written to the log file TCPWARE:NFSSERVER.LOG. This parameter is a bit mask value (in decimal), as defined in Table 13-5.

You cannot disable fatal errors and the Server writes them to OPCOM. The default (`-1`) is all classes of information enabled.

Table 13-5   NFS_LOG_CLASS Bit Mask Values

Bit...	Means when set...	Which are...
1	Warnings	Error recovery messages
2	MOUNT requests	MOUNT call messages
4	General	General operation messages
8	Security	Security violation messages
16	NFS errors	NFSERR_IO messages
(remaining)		Reserved for future use

NFS_PCNFSD_ENABLE (1)	Enables or disables the PCNFSD services support. A value of 1 enables the PCNFSD services support. A value of 0 disables the support. A value of 3 enables print spooling of files on the server without enabling PCNFSD authentication. The logical name for NFS_PCNFSD_ENABLE is TCPWARE_PCNFSD_ENABLE.
NFS_PCNFSD_SPOOL	Specifies the name of the PCNFSD print spool directory as a UNIX style pathname. The directory must be an exported directory. This is, the directory must be an entry in the EXPORT database, or a subdirectory of an exported directory. The logical name for NFS_PCNFSD_SPOOL is TCPWARE_PCNFSD_SPOOL. If the path specifies a subdirectory of an exported directory, each path element below the exported directory must match the corresponding OpenVMS subdirectory name. The filename translation rules, as described in Appendix A, *NFS-to-OpenVMS Filename Mapping*, apply to the path elements below the export point. ***Note!*** Because you export different OpenVMS directories to different clients with the same path, it is possible for the NFS_PCNFSD_SPOOL parameter to refer to different OpenVMS directories depending on which PCNFSD client requests the print spooling services.

Advanced Parameters

You should not normally change the parameters described in this section. If you need to change a value for an advanced parameter, edit the TCPWARE_SPECIFIC:[TCPWARE]TCPWARE_CONFIGURE.COM file.

The advanced parameters that follow appear in alphabetical order. The default setting for each parameter is in parentheses.

Parameter	Description
NFS_DFLT_UID (-2), **NFS_DFLT_GID** (-2)	Specifies the default UID and GID. The Server uses these defaults in the following cases: • The server receives a request from a user without a PROXY mapping and who is also the superuser (UID=0, and any GID). The Server replaces the superuser UID and GID with the default UID and GID. • The server processes a `get attributes` request and cannot find a file's owner UIC in the PROXY database. The Server uses the default UID and GID instead.
NFS_DIRLIFE_TIMER (:3)	Sets when to delete internal directory cache data structures. The Server periodically scans these data structures and deletes them if a directory's cache has existed for longer than the NFS_DIRLIFE_TIMER value. This preserves memory. Specify the interval as OpenVMS delta time. The default is 3 minutes. If you are unfamiliar with delta time, see Chapter 12, *NFS-OpenVMS Client Management*, *Client Commands*.

Parameter	Description
NFS_DIRREAD_LIMIT (`-1`)	Sets the maximum size in bytes for each file read while processing a `get attributes` request. If the estimated file size exceeds this value, TCPware does not read the file to determine its exact size and returns an estimated size instead. The estimated file size is always larger than the exact size. The `-1` default effectively turns off file size estimation. This parameter applies only to filesystems exported with the /CONVERT option (the default). A value of `0` disables TCPware from determining exact file sizes on requests. This parameter may provide the NFS Client with inexact file sizes. This is generally not a problem, but may affect some applications.
NFS_DIRTIME_TIMER (`::30`)	Sets a time interval that determines when the Server updates the directory access time between NFS operations. Specify the interval as an OpenVMS delta time. The default is 30 seconds.
NFS_FILE_CACHE_SIZE (`1024`)	Determines the maximum number of files allowed to have attributes in cache at any one time. The number must be larger than the SYSGEN parameter CHANNELCNT. The value must also be larger than the number of combined TCP and UDP threads (see the NFS_TCP_THREADS and NFS_UDP_THREADS parameters).
NFS_NOCHECKSUM (`0`)	Enables or disables checksum generation for UDP datagrams. This parameter is a boolean value. When the value is 0 (false), the Server generates checksums for outgoing datagrams. When the value is 1 (true), the Server does not generate checksums. Enabling checksums maintains data integrity, and is the default. Note: Disabling checksums may increase system performance but could have an adverse affect on certain NFS clients.

Parameter	Description
NFS_OPENFILE_TIMER (::6)	Sets a time interval (in delta time) a file remains open after you last accessed it. This can speed up request processing since a file can remain open for successive read or write requests. You do not need to open and close it for each request. The default is six seconds. You should not leave a file open for extended time, nor leave it open for too short an interval, which can decrease performance.

The following parameters are only meaningful if PCNFSD was enabled during TCPware installation:

Parameter	Description
NFS_PCNFSD_DFLTPRTOPT	Specifies the default print options when submitting a spooled print job for printing. The TCPware logical name for NFS_PCNFSD_DFLPRTOPT is TCPWARE_PCNFSD_DFLTPRTOPT.
NFS_PCNFSD_JOB_LIMIT	Specifies the maximum packet size of the information displaying the queued print jobs. Some systems require this limitation. Note that if the actual queued job information exceeds the byte limit set by this parameter, TCPware truncates the information. The TCPware logical name for NFS_PCNFSD_JOB_LIMIT is TCPWARE_PCNFSD_JOB_LIMIT. If you do not define this logical, TCPware determines the size of the packet at run-time.
NFS_PCNFSD_PRINTER (`SYS$PRINT`)	Specifies the print queue you want used if the NFS client does not specify a printer. This is an optional parameter and the default is SYS$PRINT when the client does not specify a printer (most clients specify the printer). The TCPware logical name for NFS_PCNFSD_PRINTER is TCPWARE_PCNFSD_PRINTER.
NFS_PCNFSD_PRINTER_LIMIT	Specifies the maximum packet size of the information displaying the printers known on the server. Some systems require this limitation. Note that if the actual printer information exceeds the byte limit set by this parameter, TCPware truncates the information. The TCPware logical name for NFS_PCNFSD_PRINTER_LIMIT is TCPWARE_PCNFSD_PRINTER_LIMIT. If you do not define this logical, TCPware determines the size of the packet at run-time.

Parameter	Description
NFS_PORT (`2049`)	Sets the TCP and UDP port through which the NFS, MOUNT, and PCNFSD protocols receive data.
NFS_TCP_THREADS (`20`)	Controls the number of simultaneously serviced requests received over TCP connections the Server can support. The Server requires a thread for each TCP request it receives. This thread is active for the amount of time it takes the server to receive the request, perform the operation, and send a reply to the client. The more threads the Server supports, the better the performance, because the Server can process more requests simultaneously. Note that the number of threads has no impact on the number of TCP connections the Server supports.
NFS_UDP_THREADS (`20`)	This is similar to the NFS_TCP_THREADS parameter but relates to UDP threads.

Parameter	Description
NFS_XID_CACHE_SIZE (40)	Sets the maximum number of XID cache entries. The XID cache stores replies to requests for all NFS protocol operations. When the server receives a request for an operation, it checks the XID cache for a reply to the same request. If the server locates the reply, it retransmits it. If it cannot locate the reply, it processes the request normally. The XID cache prevents the system from transmitting false error messages for operations such as delete, create, rename, and set attributes. When TCPware receives a request for one of these operations, it checks the XID cache for a reply to the same request. If the reply exists, the server retransmits it. For example, the Server receives a delete file request from a remote host. After the Server deletes the file and sends a success reply, the network loses the reply. Because the remote host does not receive a reply, it sends the delete file request again. Without an XID cache, TCPware would try to process the request again and send a false error message that it could not find the file. The XID cache prevents the system from sending the false error because it stores and retransmits the original reply. Set the NFS_XID_CACHE_SIZE parameter to at least twice (2 times) the largest of the number of: • NFS clients using the NFS Server • UDP threads (as set by the NFS_UDP_THREADS parameter) • TCP threads (as set by the NFS_TCP_THREADS parameter) The parameter sets the size of both the UDP and TCP XID caches (each protocol has a separate XID cache).

Implementation

This section describes the Server restrictions and implementation of the Network File System (NFS) protocol. The material presented here requires a thorough understanding of the protocols. It does not explain or describe the protocols.

Restrictions

The Server has the following OpenVMS-related restrictions:

- The Server supports Files-11 ODS-2 structure level disks and any CD-ROM format.
- The Server does not implement volume protection. All exported devices should be public devices.
- The Server does not generate security or audit alarms. However, the Server writes access violations to log file TCPWARE:NFSSERVER.LOG (as long as you enable security logging through the NFS_LOG_CLASS parameter).
- When creating files and directories, the Server sets the owner UIC of the file or directory to the UIC derived from the UID/GID in the create request authentication information or to the UID/GID in the set attributes information (if available).

NFS Protocol Procedures

The Server implements the following NFS protocol (version 2) procedures:

Procedures	Description
CREATE FILE (`create`)	The Server creates files using the record format specified in the EXPORT database entry. If a file of the same name exists, the Server overwrites the existing file unless the filename version number is zero or negative.

Procedures	Description
GET ATTRIBUTES (`getattr`)	Gets a file's attributes. The Server handles certain file attributes in ways that are compatible with the OpenVMS system. These attributes are: **File protection**--The Server maps the OpenVMS file protection mask to the UNIX file protection mask. **Number of links**--Although OpenVMS supports hard links, it does not maintain a reference count. Therefore, the Server sets this value to 1 for regular files and 2 for directory files. **UID/GID**--The Server maps a file owner's UIC to a UID/GID pair through the PROXY database. **Device number**--The Server returns the device number as -1. **Number of blocks**--The number of blocks returned is the number of blocks allocated to the file. **Filesystem id**--The Server returns the filesystem ID as 0. **Access, modify, status change times**--The OpenVMS system does not maintain the same file times as NFS requires. The Server returns the OpenVMS revision (modify) time for all three NFS times. For directory files, the Server returns the access, status change, and modify times as a reasonably recent time, based on the time of the last Server-initiated directory change, and the NFS_DIRTIME_TIMER parameter. This is a benefit to clients that cache directory entries based on the directory times. OpenVMS bases its time on local time, while UNIX bases its time on Universal time (or Greenwich mean time), and these times may not agree. The offset from Universal time specified when configuring TCPware resolves the difference between local and Universal time.
GET FILESYSTEM STATISTICS (`statfs`)	Returns filesystem statistics. The Server handles certain file attributes in ways that are compatible with the OpenVMS system. These attributes are: **Block size**--The block size is 1024. **Total number of blocks**--The total number of blocks is the SYS$GETDVI MAXBLOCK parameter divided by 2. **Blocks free**--The number of blocks free is the SYS$GETDVI FREEBLOCK parameter divided by 2. **Blocks available**--The number of blocks available to unprivileged users is the same as the number of blocks free.
LINK (`link`)	Creates a hard link to a file. The Server stores the link count in an application access control entry (ACE) on the file.

Procedures	Description
LOOKUP FILE (`lookup`)	Looks up a file name. If the file name does not have a file extension, the Server first searches for a directory with the specified name. If the Server fails to locate a directory, it searches for the file name without an extension.
MAKE DIRECTORY (`mkdir`)	Creates a directory. The OpenVMS system does not allow the remote host to create more than eight directory levels from the root of the OpenVMS filesystem. The Server ignores access and modify times in the request.
READ DIRECTORY (`readdir`)	Reads a directory. The Server returns file names using the filename mapping scheme as specified in the EXPORT database entry. The Server also drops the VMS version number from the file name for the highest version of the file.
READ FROM FILE (`read`)	Reads from a file. The Server converts VARIABLE and VFC files to STREAM or STREAM_LF format (depending on the option set) as it reads them.
REMOVE DIRECTORY (`rmdir`)	Deletes a directory.
REMOVE FILE (`remove`)	Deletes a file.
RENAME FILE (`rename`)	Renames a file. If the destination filename is the same as an existing filename and the destination filename does not have a zero or negative version number, the Server overwrites the existing file.
READ LINK (`readlink`)	Reads the contents of a symbolic link.

Procedures	Description
SET ATTRIBUTES (`setattr`)	Sets file attributes. The Server handles certain file attributes in ways that are compatible with the OpenVMS system. These attributes are: **File protection**--The Server maps the UNIX file protection mask to the OpenVMS file protection mask, as shown earlier in this chapter. **UID/GID**--The client changes the file owner's UIC. The PROXY database maps the new UID/GID to an OpenVMS UIC. If the Server cannot locate the new UID/GID in the database, it returns an error and does not change the owner UIC. **Size**--If the file size is larger than the allocated size, the Server extends the file. If the size is 0, the Server truncates the file and sets the record attributes to sequential STREAM_LF. You cannot change the size of variable length or VFC files (except to zero). **Access time**--Changing the access time has no effect on the OpenVMS system. **Modify time**--The modify time updates the OpenVMS revision time.
SYMBOLIC LINK (`symlink`)	Creates a symbolic link. The Server creates the file with an undefined record structure and uses an application ACE on the file to mask it as a symbolic link.
WRITE TO FILE **(`write`)**	Writes to a file. The Server does not allow a remote host to write to a directory file, or to VARIABLE and VFC files. If the Server allowed a remote host to write to an existing OpenVMS file that was not a STREAM_LF or fixed-length record format file, the file could become corrupted. The Server does not allow a remote host to explicitly change the record format of an OpenVMS file. The Server can return the non-standard NFS error ETXTBSY (26) and EINVAL (22). The Server returns ETXTBSY when an OpenVMS user has a file open for exclusive access and an NFS user tries to use the file in a way that is inconsistent with the way the OpenVMS user opened the file. The Server returns EINVAL if an NFS user tries to write to or change the size of a VARIABLE or VFC record format file.

PCNFSD Protocol Procedures

The NFS Server implements both the PCNFSD Version 1 and Version 2 protocol procedures, offers printer support, and offers additional break-in security.

PCNFSD Version 1

The PCNFSD Version 1 procedures include:

AUTHENTICATE	Performs user authentication. Maps a username and password into a UID/GID pair from the PROXY database.
INITIALIZE PRINTER	Prepares for remote printing. Returns the pathname of the client's spool directory. The Server concatenates the spool directory path (derived from NFS_PCNFSD_SPOOL parameter) with the client name.
NULL	The null procedure; standard for all RPC programs.
START PRINTING	Submits a spooled print job for printing. The print data is in a file created in the spool directory, which the Server identifies by the client name. If the user omits a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. (See *Print Options*.)

PCNFSD Version 2

The supported PCNFSD Version 2 procedures include:

ALERT OPERATOR	Sends a message to the system operator. If the user does not specify a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. You cannot use batch queues.
AUTHENTICATE	Performs user authentication. Maps a username and password into a UID/GID pair from the PROXY database.
CANCEL PRINT	Cancels a print job. If the user does not specify a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. You cannot use batch queues.
HOLD PRINT	Places a hold on a previously submitted print job. The job remains in the queue but the Server does not print it. If the user does not specify a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. You cannot use batch queues.
INFORMATION	Determines which services the current PCNFSD implementation supports.
INITIALIZE PRINTER	Prepares for remote printing. Returns the pathname of the client's spool directory. The Server concatenates the spool directory path (derived from NFS_PCNFSD_SPOOL parameter) with the client name.

LIST PRINTERS	Lists all printers known on the server, except if the NFS_PCNFSD_PRINTER_LIMIT parameter sets the packet size less than the actual amount of information.
LIST QUEUE	Lists all or part of the queued jobs for a printer, depending on how you set the NFS_PCNFSD_JOB_LIMIT parameter.
NULL	The null procedure; standard for all RPC programs.
PRINTER STATUS	Determines the status of a printer. If the user does not specify a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. You cannot use batch queues.
RELEASE PRINT	Releases the "hold" on a previously held print job. If the user does not specify a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. You cannot use batch queues.
START PRINTING	Submits a spooled print job for printing. The print data is in a file created in the spool directory, which the Server identifies by the client name. If the user does not specify a printer, the Server uses the default printer set by the NFS_PCNFSD_PRINTER parameter. The Server submits the job using the print options described next.

Print Options

The NFS Server submits the job using the print options in Table 13-7.

Table 13-6 Print Options Settings

Print option...	**Has values...**	
r (RFM type)	**u** (undefined), **s** (stream), **l** (streamlf), **c** (streamcr), **n** (none)*****	
f (file flag)	+ (flag)	**-** (no flag)*****
e (paginate)	+ (feed)	**-** (no feed)*****
h (page header)	+ (header)	**-** (no header)*****
s (double space)	+ (double space)	**-** (no space)*****
b (file burst)	+ (burst)	**-** (no burst)*****
t (file trailer)	+ (trailer)	**-** (no trailer)*****
p (log spool)	+ (log spool)	**-** (no log spool)*****
l (passall)**	+ (passall)	**-** (no passall)*****
c (number of copies)	character string in range from 1 to 255, otherwise=1*****	
***** Defaults set by PCNFSD; when defined, the NFS_PCNFSD_DFLTPRTOPT parameter overrides any or all of the default values ****** If you use the **l+** option (passall), TCPware ignores all the other options		

The following steps show the procedure and syntax for setting up an NFS_PCNFSD_DFLTPRTOPT parameter using several of the available print options:

1	Edit the TCPWARE_SPECIFIC:[TCPWARE]TCPWARE_CONFIGURE.COM file.
2	Find the line containing the definition for the NFS_PCNFSD_DFLTPRTOPT parameter in the code. It originally appears in the TCPWARE_CONFIGURE.COM file as follows: `$ NFS_PCNFSD_DFLTPRTOPT == ""`
3	Modify the line to specify the desired print options in the quotation marks. The syntax is as in the following example: `$ NFS_PCNFSD_DFLTPRTOPT == "h+t+rl"`

4	Shut down and restart NFS as follows:
	`$ @TCPWARE:SHUTNET NFS`
	`$ @TCPWARE:STARTNET NFS`

This example submits the print job using a page header, a file trailer, and STREAMLF record format type. This example also uses the remaining print option defaults.

The PC-NFS client can further override the print options default values. These print options specified by the PC-NFS Client are only relevant for the filename specified in the PC-NFS client request packet.

Break-in Security

PCNFSD uses the OpenVMS Intrusion database to store intrusion records, unless disabled during NFS-OpenVMS Server configuration. When the PC sends an invalid user authentication request to the NFS Server, the Server checks the Intrusion (break-in) database. The database indicates the number of invalid mount requests that exceeds the threshold set for detecting break-in attempts.

If the NFS Server reaches the threshold number of invalid mount requests, it logs this as an attempted break-in. This locks out the PC until you remove the intrusion record or through other ways described in Compaq's *Guide to System Security.*

You can show intrusions by using the SHOW INTRUSION/OLD command at the DCL level. You can then remove any offending entries by using the DELETE/INTRUSION_RECORD *source* command on the DCL level. The *source* parameter is the remote device or system where the user tries to log in. (Both commands require the SECURITY privilege.)

See the SHOW INTRUSION/OLD and DELETE/INTRUSION_RECORD commands in Compaq's *VMS DCL Dictionary* for details.

Troubleshooting

If you are experiencing network communication-related problems on the NFS-OpenVMS Server, please check the following items:

1	Make sure TCPware is running on the OpenVMS system.
2	Make sure the Server is running. If not, start it by entering the following command at the DCL prompt: `@TCPWARE:STARTNET NFS` If the Server is not running but was started, examine the TCPWARE:NFSSERVER.LOG file. This file contains information to help you isolate problems with the Server. After correcting any problems that were reported in the log file, restart the Server.
3	To verify general connectivity between the two systems, try using FTP or TELNET, if purchased and installed on your system. For example, try to open a TELNET connection with the remote host in question. If another TCPware product is not available on your system, try using the TCPware PING utility.
4	Verify the internet addresses the local host and the remote hosts are using. If your local network includes a gateway, also verify the gateway address.

If you are experiencing problems performing NFS operations from a NFS client, check the Server's TCPWARE:NFSSERVER.LOG file. It may contain messages that can help isolate the problem.

Certain messages can also come up with the NETCU SHOW EXPORT, SHOW MOUNT, and UNMOUNT commands.

Access error messages help by entering **`HELP TCPWARE MESSAGES`**.

Chapter 14

Managing Print Services

Introduction

This chapter describes how to manage the TCPware print services, which include the Line Printer Services (LPS) client and server, and the Terminal Server Print Services (TSSYM).

Line Printer Services Client

You can configure an OpenVMS host with both an LPS client and a server. The LPS client lets users send print jobs to printers attached to remote hosts. It supports the UNIX-like LPR commands and the OpenVMS PRINT command. You can configure the LPS client to use:

UNIX-style LPR commands (`lpr`, `lpq`, and `lprm`)	During configuration, enter information about the default remote host and printer when you use an LPR command.
PRINT	Command used with one or more OpenVMS print queues on the client. TCPware creates and starts these queues during STARTNET. These queues can use: • OpenVMS /FORM features on the local or remote print queues. • The /PARAMETERS qualifier to achieve minimal formatting on remote print queues.

You can set up the print queues during TCPware configuration, or base the settings on entries in a local PRINTCAP (printer capability) database.

The PRINT command supports two different print symbionts:

TCPWARE_VMSLPRSMB	Provides local print queue formatting

TCPWARE_LPRSMB	Provides remote print queue formatting

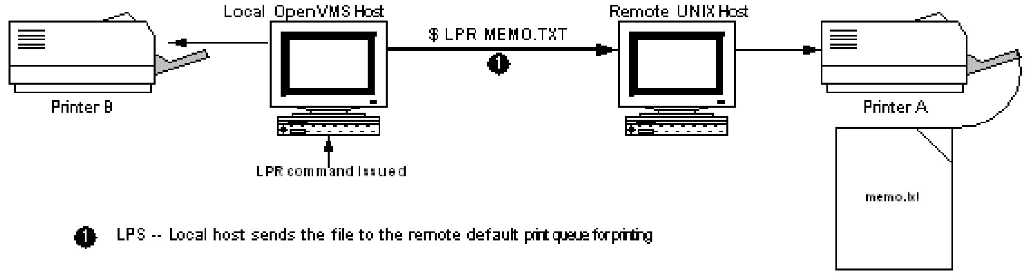
LPS Client Commands

The LPS client supports the following commands:

Command	Description	Command	Description
LPQ	Displays the remote print job status	LPRM	Removes a job from a remote print queue
LPR	Sends a job to the default remote printer designated during configuration	PRINT	Places a job in the designated print queue; then sends the job to the printer associated with that queue.

Figure 14-1 shows how to use an LPR command with an LPS configuration.

Figure 14-1   Using a UNIX-Style LPR Command



OpenVMS Print Queues

The LPS client uses LPS print symbionts to control where document formatting and device control occurs: locally or remotely. The two symbionts (shown in Table 14-1) provide different kinds of print format support, configurable during the CNFNET procedure.

Table 14-1   LPS Print Symbionts

Symbiont...	Supports the command...	And forms definition...
TCPWARE_VMSLPRSMB (provides local print formatting control)	PRINT/FORM	All DEFINE/FORM qualifiers

Table 14-1 LPS Print Symbionts (Continued)

Symbiont...	Supports the command...	And forms definition...
TCPWARE_LPRSMB (provides remote print formatting control)	PRINT/PARAMETERS and some PRINT/FORM	DEFINE/FORM /SETUP /STOCK /DESCRIPTION only

See the *User's Guide*, Chapter 5, *Network Printing*, and your OpenVMS documentation for information about the PRINT command and its qualifiers.

You can also configure the OpenVMS print queues to support additional qualifiers available with the OpenVMS INITIALIZE /QUEUE command. Use the /LIBRARY=LN03DEVCTL qualifier to enable the device control library. The device control library contains device control modules and resides in SYS$LIBRARY.

Print Forms

You must configure the print forms specifically for LPS to control the local or remote printer's setup for each print job. Use the OpenVMS DEFINE /FORM command format, with the qualifiers listed in Table 14-2, as supported for each of the print symbionts:

DEFINE/FORM *form-name* *form-number* *qualifier* ...

Table 14-2 Supported DEFINE/FORM Qualifiers

DEFINE/FORM qualifier...	Description
/LENGTH=*n* /MARGIN= (BOTTOM=*n*, LEFT=*n*, RIGHT=*n*, /TOP=*n*) /PAGE_SETUP=*module* /SHEET_FEED, /TRUNCATE, /WIDTH=*n*, /WRAP	Sets various page setup print parameters. (Use with TCPWARE_VMSLPRSMB only.)
/SETUP=*module*	Device control module or modules (separated by commas) that contain the print control sequences for the remote printer. (Use with TCPWARE_VMSLPRSMB or TCPWARE_LPRSMB.)

Table 14-2 Supported DEFINE/FORM Qualifiers (Continued)

DEFINE/FORM qualifier...	Description
/STOCK=*string*	Type of paper stock to associate with the form (if not the same as the form-name); *string* can be up to 31 characters. (Use with TCPWARE_VMSLPRSMB or TCPWARE_LPRSMB.)
/DESCRIPTION=*string*	Operator information about the form (the default is the form-name) that appears when you issue the SHOW QUEUE/FORM command; use quotes to preserve case or if using spaces; *string* can be up to 255 characters. (Use with TCPWARE_VMSLPRSMB or TCPWARE_LPRSMB.)

PRINTCAP Database

During TCPware configuration, you can select whether to use the PRINTCAP (printer capability) database, if it exists, to start your local OpenVMS queues for the PRINT command. (Make any subsequent queue definitions during the usual LPS configuration.) The PRINTCAP database is the equivalent of the UNIX `/etc/printcap` file and resides locally in the `TCPWARE:PRINTCAP.` file. If you do not have or opt not to use the PRINTCAP database to define local print queues, you must define these queues one by one during configuration.

The PRINTCAP database requires a special syntax. Each entry in the database describes one printer. According to the UNIX convention, each entry in the file is one or more lines consisting of fields separated by colon (**:**) characters. The first entry for each printer gives the name or names under which the printer is known, separated by vertical bar (**|**) characters. Entries can continue onto multiple lines by adding a backslash (\) after the last character of a line. You can include empty fields for readability.

You can use a number of boolean, numeric, and string type options (or *capabilities*) with each database entry, although TCPware only supports three string type capabilities:

lp	Local printer's device name
rm	Remote machine name
rp	Remote printer name

An equal sign (**=**) separates the capability code from its value.

Example 14-1 shows a sample entry in the `PRINTCAP.` file.

Example 14-1 Remote Print Queue Names in the Local PRINTCAP. File

```
#
# LOCAL PRINTERS
#
test_printer:
  :lp=test_printer:
  :rm=alcor:
  :rp=eng2_printer:
```

In this example, the name of the local printer (`lp`) is `test_printer`. The remote machine (`rm`) is `alcor`, and the remote printer (`rp`) is `eng2_printer`. Lines starting with the pound sign (`#`) and blank lines are ignored.

To print to TEST_PRINTER, users specify:

```
$ PRINT/QUEUE=TEST_PRINTER filespec
```

The output appears on node ALCOR's ENG2_PRINTER printer.

Note! The PRINTCAP database is not dynamic. To institute any changes you make to it, you must reconfigure the OpenVMS print queue using the configuration procedure.

LPS System Logicals

LPS uses several system logicals. TCPware defines only those LPS logicals required for features that you enable during CNFNET in the TCPWARE_SPECIFIC:TCPWARE_

CONFIGURE.COM file. STARTNET uses the information in this file to create the logicals when you start the network. For example, TCPware defines logicals related to the LPD server only if you enable the server during CNFNET. Change features that you enable by rerunning CNFNET.

After you start the network, use the SHOW LOGICAL command in OpenVMS to examine the logical definitions. To set up a generic LPS client queue to print to a machine, set up the TCPWARE_LPR_*qname*_PRINTER logical for both the generic and server queues. The server queue automatically sets up the logical after you define it.

Client Logicals

Table 14-3 explains the purpose of each LPS client logical.

Table 14-3 LPS Client Logicals

LPS logical...	Description
TCPWARE_LPR_QUEUES	Lists the names of all TCPware print symbiont queues. Defined only if you defined one or more print queue.

Table 14-3 LPS Client Logicals (Continued)

LPS logical...	Description
TCPWARE_LPR_*qname*_ PRINTER	Defines the absolute printer for the PRINT command. You cannot override this logical when submitting a print job. Use to restrict printing to one printer per queue.
TCPWARE_LPR_*qname*_ PRINTER_DEFAULT	Defines a default remote printer for the PRINT command. Used if neither TCPWARE_LPR_*qname*_PRINTER nor the PRINT command specify a remote printer. You must define either TCPWARE_LPR_*qname*_PRINTER or TCPWARE_LPR_*qname*_PRINTER_DEFAULT for each queue for the PRINT command.
TCPWARE_LPR_SPOOL	Points to the work directory for the PRINT command. This directory holds temporary files.
TCPWARE_LPR_PRINTER	Defines the default remote printer for the LPR, LPRM, and LPQ commands. You can define your own TCPWARE_LPR_PRINTER logical in a LOGIN.COM file.

VMSLPRSMB Tuning Logicals

The TCPWARE_VMSLPRSMB print symbiont provides the retry interval and timeout tuning logicals (all are executive mode system logicals) listed in Table 14-4.

Table 14-4 VMSLPRSMB Tuning Logicals

VMSLPRSMB logical...	Description
TCPWARE_VMSLPRSMB_*_ RETRY_INTERVAL	Defines the interval at which the symbiont retries to make a connection to a printer after an attempt fails. The connection can fail either by the remote printer rejecting it due to a busy state, or by the network timing out. The default value for a retry interval is 2 minutes (:2 in delta time). Note that a connection failure can take 1.5 minutes to time out, which is not included in this interval value.

Table 14-4 VMSLPRSMB Tuning Logicals (Continued)

VMSLPRSMB logical...	**Description**
TCPWARE_VMSLPRSMB_ *qname*_RETRY_INTERVAL	Same as TCPWARE_VMSLPRSMB_*_RETRY_INTERVAL, but for a specific queue only, and overrides TCPWARE_VMSLPRSMB_*_RETRY_INTERVAL.
TCPWARE_VMSLPRSMB_*_ TIMEOUT	Defines the time it takes for a print job to abort if the connection to the printer is never established. The default timeout is infinite (it never times out).
TCPWARE_VMSLPRSMB_ *qname*_TIMEOUT	Same as TCPWARE_VMSLPRSMB_*_TIMEOUT, but for a specific queue only, and overrides TCPWARE_VMSLPRSMB_*_TIMEOUT.
TCPWARE_VMSLPRSMB_ *qname*_PRECONN	Makes the connection to the printer *before* processing the file. Normal behavior is to make the connection to the printer *after* processing the file.

LPRSMB Tuning Logicals

The TCPWARE_LPRSMB print symbiont provides similar retry interval and timeout tuning logicals as those for TCPWARE_VMSLPRSMB (see the descriptions in Table 26-4). The TCPWARE_LPRSMB logicals are:

- TCPWARE_LPRSMB_*_RETRY_INTERVAL
- TCPWARE_LPRSMB_*qname*_RETRY_INTERVAL
- TCPWARE_LPRSMB_*_TIMEOUT
- TCPWARE_LPRSMB_*qname*_TIMEOUT
- TCPWARE_LPRSM_*qname*_PRECONN

Troubleshooting LPS

You may have a SETUP module in your LPS print queues (TCPWARE_VMSLPRSMB or TCPWARE_TSSYM) that causes the OpenVMS print symbiont to insert unexpected form feeds (`<FF>`). (See *Print Forms*.)

You can remove these form feeds by adding an undocumented escape sequence to the SETUP module, as follows:

1	Start the SETUP module with the special escape sequence: **`<ESC>]VMS;2<ESC>`**

2	Enclose the setup text with: **<ESC>P*string*<ESC>** For example, if you want to send the setup text setup to the printer, the SETUP module could look like this (or you could have two setup modules, one with the <ESC>]VMS;2<ESC>F0> text and the other with the <ESC>Psetup<ESC>F0> text): **<ESC>]VMS;2<ESC>Psetup<ESC>F0>**

LPD Server

The LPD server on the local host accepts print requests from remote users. It then places the remote files in local OpenVMS print queues. You must define and initialize these OpenVMS print queues before you configure the TCPware LPD server.

Sending files from remote UNIX systems to a local OpenVMS printer requires the UNIX system to have an entry in an /etc/printcap file. Some UNIX systems do not have this file and rely on another method. (See your UNIX documentation for more information.) Here is a sample entry in an /etc/printcap file:

```
rpl | remote printer:
:lp=:
:sd=/usr/spool/lpd:
:rm=daisy:
:rp=ln03r$print
```

The following UNIX command puts myfile in the ln03r$print queue on daisy:

```
lpr -Prpl myfile
```

Server Supported Options

The LPD server supports the options listed in Table 26-5. (It does not support other options and ignores print requests you issue with such options, without issuing an error message.)

Table 14-5 Options Supported by LPD Server

For command...	This option...	Does...
LPQ	-l	Displays the status of each job on more than one line
LPR	-C	Prints the job classification on the burst page (like the PRINT/NOTE command in OpenVMS)

Table 14-5 Options Supported by LPD Server (Continued)

For command...	This option...	Does...
	-f	Interprets the first character of each line as a standard FORTRAN carriage control character
	-h	Prevents the burst page from printing (like the PRINT/NOFLAG command in OpenVMS)
	-J	Prints the job name on the burst page (like the PRINT/NAME command in OpenVMS)
	-l	Prints control characters and suppresses page breaks (like the PRINT/PASSALL/NOFEED/NOHEADER command in OpenVMS)
	-m	Notifies the OpenVMS user when printing has completed for the job (like the PRINT/NOTIFY command in OpenVMS)
	-o	Indicates the file contains PostScript input
	-p	Prints the file with page headers (like the PRINT/HEADER command in OpenVMS)
	-v	Prints the Sun raster format file as a binary file with no formatting
	-x	Specifies not to require filtering before printing (like the PRINT/PASSALL/NOFEED/HEADER command in OpenVMS)
	-#	Prints multiple copies (like the PRINT/COPIES command in OpenVMS)
LPRM	-	Removes all jobs that only you own

Data and Control Files

The LPD server accepts only data files and control files from clients. Data files contain copies of the data you want printed or executed. Control files store the commands that specify how you want the data printed.

LPD receives the data and control files in STREAM-LF format unless you use **`lpr-l`** to send the print job to the printer. It stores the files in the spooling directory until the job ends. The TCPWARE_LPD_SPOOL logical points to the spooling directory.

LPD Access File

The LPD server requires an LPD access file. It checks this file before accepting any requests from remote clients. This file:

- Determines which remote hosts can access the local LPD server.
- Maps remote users to OpenVMS usernames.

You can create the LPD access file during or after TCPware configuration. Use any text editor to enter data in the file. If you create the file after configuring TCPware, give it the name TCPWARE_COMMON:[TCPWARE]LPD_USERS.DAT. Use the following format:

vms-username remote-host remote-user

vms-username	Username defined in the OpenVMS User Authorization File. Upper- or lowercase characters are acceptable. You can enter an asterisk (*) as a wildcard to designate the incoming user as the *vms-username*. Use a hyphen (-) to specifically disallow access to printing services.
remote-host	Host on which the remote user resides. Enter the full or partial domain name, or the internet address. (Using the internet address improves performance.) Upper or lowercase characters are acceptable. You can enter an asterisk (*) as a wildcard to designate all remote hosts. Do not partially wildcard the host name.
remote-user	Username on the remote host. Enter the username in the same case (upper- or lowercase) as the remote host uses to define it. You can enter an asterisk (*) as a wildcard. The wildcard maps all remote users to this *vms-username* account entry.

CAUTION! Use wildcards cautiously. They can cause severe security problems.

Use the exclamation point (!) or pound sign (#) as the first character of a line to indicate a comment line.

Include at least one space or tab between each item.

When the remote user tries to access the LPD server, LPD looks at LPD_USERS.DAT for valid username mapping. If a valid username mapping is not found, LPD checks the system logical TCPWARE_LPD_DEFAULT_USER to determine the OpenVMS username. If this system logical is not found, the LPD server discards and never prints the file. You define the OpenVMS username for this logical during network configuration.

When LPD receives a job from a remote system, the format of the print job’s **`NOTE`** is:

remote-ID: *user@host - note*

- ***user*** – Remote username
- ***host*** – Remote hostname
- ***note*** – Note as specified on the LPR command, or the default (often the hostname)

Here is a sample LPD access file:

```
!vms-username    remote-host     remote-user
!-------------------------------------------
smith            daisy           smith
jones            daisy           jones
jones            rose            jones
harrington       192.168.95.1    harrington
harrington       tulip           harrington
wallace          *               wallace
harrington       iris            *
```

It is recommended you place wildcard entries later in the file, as the first acceptable mapping will be used.

The following example illustrates an access file which provides a specific mapping for remote user gertrude. It allows access to all users with matching names on both systems, and provides a default mapping for all other users on node daisy.

```
!vms-username   remote-host   remote-user
!-----------------------------------------
-               daisy         thorn
rose            daisy         gertrude
*               daisy         *
daisy_default   daisy         *
```

In the first line, user thorn on system daisy is denied access to printing services.

In the second line, the remote user gertrude on daisy is mapped to the OpenVMS username rose.

In the third line, the LPD server is instructed to map, as is, usernames having corresponding OpenVMS accounts.

In the fourth line, if the remote user on daisy does not have a corresponding OpenVMS account on the local system, it is mapped to account DAISY_DEFAULT.

Batch Queues

The LPD server can place jobs in batch queues for execution. You enable this feature during network configuration. To send a job to a batch queue, specify the batch queue name instead of the print queue name when you enter the PRINT or LPR command.

The LPD server does not support qualifiers or options that are analogous to the following OpenVMS SUBMIT command qualifiers: /CLI, /CPUTIME, /LOG_FILE, /PRINTER, /WSDEFAULT, /WSSEXTENT, and /WSQUOTA.

LPD Logicals

See *LPS System Logicals*.

Table 14-6 explains the purpose of each LPD server logical.

Table 14-6   LPD Server Logicals

LPD logical...	Description
TCPWARE_LPD_DEFAULT_ USER	Defines a default OpenVMS username for remote users connecting to the local LPD server. Used only when you define a remote host in the LPD access file and the remote username is not mapped to a specific OpenVMS username.
TCPWARE_LPD_OPTIONS	Determines if the server handles batch queues.
TCPWARE_LPD_*qname*_ FORM*	Defines the form used for print jobs. This logical is similar to TCPWARE_LPD_*qname*_PARAMETER. Use the TCPWARE_LPD_*_FORM logical to define the form for all queues. Note that a specific queue setting overrides the global setting for that queue.
TCPWARE_LPD_*qname*_ OPTION	Specifies additional PRINT command qualifiers to pass to the specified print queue: /BURST, /FEED, /FLAG, /FORM, /HEADER, /LOWERCASE, /PASSALL, /PRIORITY, /RESTART, /SPACE, /TRAILER Use the TCPWARE_LPD_*_OPTION logical to define the option for all queues. Note that a specific queue setting overrides the global setting for that queue.
TCPWARE_LPD_*qname*_ PARAMETER*	Defines the specified parameters when the remote user submits a print request to the OpenVMS print system (*qname* is the queue name). The first equivalence string for the logical (if defined) is the first parameter; the second is the second parameter; and so on, for up to eight parameters. See Example 26-2. Use the TCPWARE_LPD_*_PARAMETER logical to define the parameter for all queues. Note that a specific queue setting overrides the global setting for that queue.

Table 14-6 LPD Server Logicals (Continued)

LPD logical...	Description
TCPWARE_LPD_*qname*_ QUEUE*	Defines the print queues for an alias queue name (*qname*). Typically supports clients that may not allow standard OpenVMS queue names as the remote printer (such as IBM's AIX, which restricts remote printer names to seven characters). See Example 26-2.
TCPWARE_LPD_SPOOL	Points to the work directory for the LPD server. This directory holds temporary files.
* STARTNET does not define the TCPWARE_LPD_*qname*_QUEUE logical. Define this system logical in the system startup file.	

Be aware of security when you define the TCPWARE_LPD_DEFAULT_USER logical. Remote users can submit batch jobs to your local OpenVMS host. To prevent unauthorized users from submitting batch jobs, do not define a username belonging to a privileged account (such as the SYSTEM username). Use AUTHORIZE instead to create a special user account with restricted access.

Example 14-2 shows how to use the TCPWARE_LPD_*qname*_PARAMETER and TCPWARE_LPD_*qname*_QUEUE logicals to support an LN03R PostScript® printer. These logical definitions define two alias queues (LP0 and PS0) for the LN03R printer queue, $LN03R1, and define the parameters for these queues. The LP0 queue prints jobs submitted as ASCII files. The PS0 queue prints jobs submitted as PostScript files.

Example 14-2 Using the LPD Logicals to Support a Printer

```
$ DEFINE/SYSTEM/EXEC TCPWARE_LPD_LP0_QUEUE $LN03R1
$ DEFINE/SYSTEM/EXEC TCPWARE_LPD_PS0_QUEUE $LN03R1
$ DEFINE/SYSTEM/EXEC TCPWARE_LPD_LP0_PARAMETER "DATA=ANSI"
$ DEFINE/SYSTEM/EXEC TCPWARE_LPD_PS0_PARAMETER "DATA=POST"
```

Troubleshooting LPD

Facilities to aid in resolving problems you may encounter with LPD include:

- OPCOM error messages (OPCOM)
- LPD log files

LPD sends messages to OPCOM under some error conditions.

Access error messages help by entering **HELP TCPWARE MESSAGES**.

LPD also writes the OPCOM messages and several other informational messages to the following LPD log file, shared by all LPD servers: TCPWARE:LPDSERVER.LOG. It is often useful to review the messages in this log file.

You can also obtain more details about LPD processing by using the Network Control Utility (NETCU) to specify an output file for SYS$OUTPUT for the LPD server. Normally, LPD's output goes to NLA0: (the null device) and is, therefore, lost. By redirecting the output to a log file, you can examine a detailed trace of LPD's execution:

```
$ NETCU MODIFY SERVICE printer TCP /OUTPUT=file
```

Terminal Server Print Services

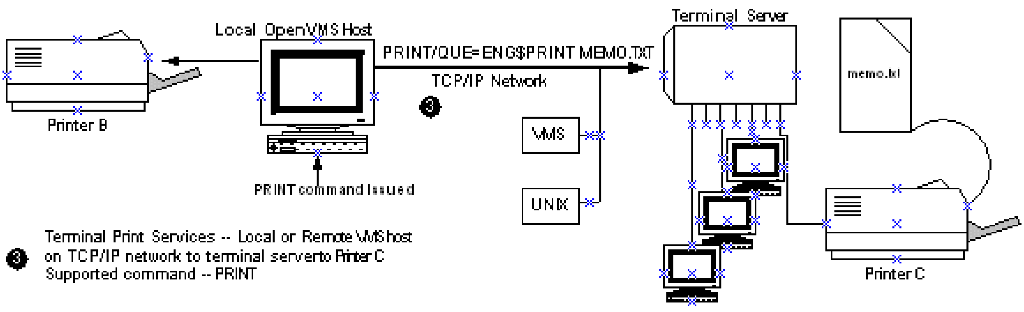
Terminal Server Print Services (TSSYM) provide an efficient way for OpenVMS users to send print requests to printers attached to TCP/IP-based terminal servers. Users on the host can easily access printers attached to a terminal server as if they were any other OpenVMS printer.

You can configure the print queues to the remote printer using standard OpenVMS print operations. Users can initiate print jobs with the usual OpenVMS commands. Figure 14-2 shows using the PRINT command with a TSSYM configuration.

A special print symbiont sends the print request over the network instead of writing it to a local printer. The symbiont performs all the necessary device-related functions on the remote terminal server. These include allocating the device, assigning a channel to it, obtaining its device characteristics, and determining its device class.

For further information on setting up print queues and initiating print commands on the OpenVMS host, see your OpenVMS documentation.

Figure 14-2 The OpenVMS PRINT Command and TSSYM



The symbiont processes the data so that the terminal server can pass the job to the printer. Unless you keep the connection open, the symbiont also closes the TCP connection to the terminal server after every print job and opens a connection with a new terminal server.

TSSYM Print Queue

Initializing the terminal server print queue requires OPER privileges. Starting the queue requires

OPER privileges or execute (E) access to the queue.

You can set up the terminal server print queue as you would any other OpenVMS printer queue, using standard OpenVMS queue commands. If you want to modify the terminal server printer queue, specify any standard printer queue commands and options the INITIALIZE/QUEUE, START/QUEUE, and SET QUEUE commands support.

When you set up the queues, the terminal server queues must specify:

/PROCESSOR=TCPWARE_TSSYM	TCPWARE_TSSYM identifies the special print symbiont.
/ON=*string*	*string* specifies the terminal server information using the *host*, *port*, *options*, and *qname* parameters (see Table 26-7).
/AUTOSTART_ON	implements an autostart queue (if enabled) on one or more nodes if used together with the TCPWARE_TSSYM_*qname* logical.

Initialize the print queue and then start it. Add the command lines shown in Table 14-7 to your printer startup commands or in your system startup file (SYSTARTUP_V5.COM or SYSTARTUP_VMS.COM).

Table 14-7 Printer Startup Commands

```
$ INIT/QUEUE/PROCESSOR=TCPWARE_TSSYM/ON="host,port[,options]" qname
$ START/QUEUE qname
```

host	With /ON, Internet address or host name (if in the HOSTS. file or resolvable using DNS) of the terminal server
port	With /ON, port number on which the terminal server accepts incoming connections for the printer port (Table 14-8 shows some common port numbers)
options	Use the following /ON options singly or in the combinations given (each option separated by a comma):

	EXPNTAB	Expands <TAB> characters to be the equivalent number of <SPACE> characters in print files. TSSYM normally ignores <TAB> characters.
	KEEP	Keeps the connection open between jobs and closes it only on errors or when exiting. Prevents several systems from sharing the printer. This eliminates opening and closing the TCP connection with every print job, thereby not tying up network resources. Do not combine with NOFF.

	`NOCRNUL`	The TELNET standard (RFC 856) requires that a carriage return character (<CR>) not followed by a line feed character (<LF>) be converted to <CR><NULL>. TCPware supports this standard. Use the NOCRNUL option to disable the character sequence conversion for printers that do not support the TELNET standard.
	`NOFF`	Does not send a form feed to the printer for each new connection. However, NOFF still sends both a form feed and a carriage return for the first file printed after you initialize the queue. Do not combine with KEEP.
	`NOINIFF`	Suppresses an initial form feed before the very first job after queue startup. Uses the sequence <CR><CR> instead of <CR><FF>.
	`NOOPCOM`	Suppresses OPCOM messages the terminal server print symbiont may produce.
	`RAW`	Makes the connection a raw, binary connection, not a TELNET connection. TCPWARE_TSSYM does not double IAC characters (ASCII 255) in the data stream. Also, <CR> is not converted to <CR><NULL>.
	`TRIMFF`	A print job normally ends with a carriage return (<CR>) and form feed (<FF>). Using the TRIMFF option, you can prevent the symbiont from adding these to the end of the print job. TRIMFF also replaces <CR> and <FF> with <CR><CR> at the beginning of the print job.
	`qname`	Name of the print queue.

Table 14-8   Common Printer Port Numbers

Printer or server...	**Is given port number...**
Emulex NetQue Print Server	2501
Emulex NetQue Serial Card	2502
HP Jet Direct Card	9100
Lantronics	20*nn*, where *nn* is the port number
Racal	100*n*, where *n* is the port number
Xylogic	70*nn*, where *nn* is the physical port on the terminal server
Xyplex 720 Terminal Server	2000 + (100 *nn*), where *nn* is port number; for example, port 14 would be 3400 for the TCP/IP listener port

Here is a typical command sequence to set up a standard ANSI printer:

```
$ INIT/QUEUE/PROCESSOR=TCPWARE_TSSYM/ON="192.168.25.50,2005" PR1
$ START/QUEUE PR1
```

If you use more than 31 characters for /ON qualifier parameters (including the quotes), the message `%QUEMAN-F-INVQUAVAL, value '"`*host*,*port*,*options*`"' invalid for /ON qualifier` appears. If you need to use more than 31 characters, define the TCPWARE_TSSYM_*qname* logical, described in *Autostart Queue*.

The standard OpenVMS qualifiers for the INITIALIZE and START commands are available. You can also set up a generic print queue where you can move print jobs to compatible execution queues, so that you can print to the first available printer on a SYS$PRINT request.

You need to start TCPware to print to the printer connected to the terminal server. If you do not start TCPWARE, the printer is down, or the system does not recognize the domain name, the print symbiont waits until you resolve the problem. This puts the print queue into a "stalled" state. In this case, you can either correct the problem while the queue is up, or stop and restart the queue using STOP/QUEUE/RESET and START/QUEUE.

Spool Device

You can set up a spool device so that you can use TSSYM to associate the device with a print queue and then perform operations such as copying a file to the device. The Compaq DATATRIEVE is an application that could use this functionality:

1	Set up a print queue, such as with the typical command sequence shown earlier.

2	Use SYSGEN to set up the spool device. Select a new device (such as `QPA0:`) and use it in the CONNECT command with the /DRIVER=FTDRIVER and /NOADAPTER qualifiers. Then specify the queue name from step 1 in the SET DEVICE /SPOOLED command, as in the following example:
	
	`$ RUN SYS$SYSTEM:SYSGEN`
	`SYSGEN> CONNECT QPA0: /DRIVER=FTDRIVER /NOADAPTER`
	`SYSGEN> EXIT`
	`$ SET DEVICE QPA0: /SPOOLED=PR1`

3	Copy a file to the device, or define the system output to be the device and run a program whose output goes to it, such as the following:
	
	`$ COPY FOOBAR QPA0:`
	`$ DEFINE SYS$OUTPUT QPA0:`
	`$ RUN PROGRAM1! output from PROGRAM1 goes to the device`

Autostart Queue

You can set up an autostart queue on a node that automatically starts up again after it stops. You can also set up such a queue to autostart on other failover nodes.

Starting an autostart queue requires the /AUTOSTART_ON qualifier for the INITIALIZE/QUEUE command. Since you cannot use /AUTOSTART_ON together with the /ON qualifier to initialize a terminal server print queue, you need to define the TCPWARE_TSSYM_*qname* logical for this purpose. This logical defines the parameters normally set with the /ON qualifier.

The format of the logical definition is:

DEFINE/SYSTEM TCPWARE_TSSYM_*qname* "*host,port[,option...]*"

The format of the /AUTOSTART_ON qualifier (use the parentheses when specifying multiple nodes):

/AUTOSTART_ON=(*node*::*[,node*::*,...]*)

Example 14-3 shows a typical command sequence to define the TCPWARE_TSSYM_*qname* logical, initialize and start up an autostart queue (QUEUE1) on two nodes, and enable autostart on these nodes. You can also add the commands to your startup command procedure. Note that there are two nodes: NODE2 can be a failover node in case NODE1 goes down.

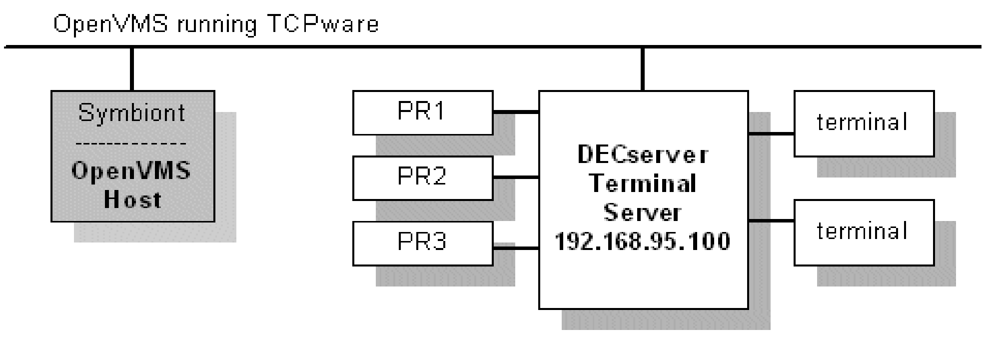
Example 14-3   Command Sequence to Set Up an Autostart Queue

```
$ DEFINE/SYSTEM TCPWARE_TSSYM_QUEUE1 "192.168.25.50,2005,KEEP"
$ INIT/QUEUE /START /PROCESSOR=TCPWARE_TSSYM -
_$ /AUTOSTART_ON=(NODE1::,NODE2::) QUEUE1
$ ENABLE AUTOSTART /QUEUES /ON=NODE1
$ ENABLE AUTOSTART /QUEUES /ON=NODE2
```

Sample TSSYM Configuration

A sample configuration includes a host connected to a DECserver 300 terminal server, as shown in Figure 14-3.

Figure 14-3   Sample TSSYM Configuration



The following example shows how to configure the OpenVMS host to process print requests to the PR1 printer on the DECserver 300. The procedure then shows how to set up the queue and execute a print request.

For details on configuring a TELNET port and the recommended settings for access through a TELNET listener for a specific printer, see Compaq's *DECserver 300 Management* guide.

The setup values in the next example are for the DECserver 300 terminal server only. For setup values specific to your terminal server, see your server documentation.

The steps in the sample configuration and startup are as follows:

<table>
<tr><td>1</td><td>Initialize the DECserver 300 terminal server.</td></tr>
<tr><td>2</td><td>At the <code>Local></code> prompt, enable privileged status to access all terminal server commands.</td></tr>
<tr><td>3</td><td>Configure the terminal server port for the printer. For example, if printer PR1 is connected to port 5:

Local> define port 5 access remote autobaud disabled
Local> define port 5 break disabled
Local> define port 5 character size 8 dedicated none
Local> define port 5 dsrlogout disabled
Local> define port 5 flow control xon inactivity logout enabled
Local> define port 5 parity none
Local> define port 5 password disabled preferred none
Local> define port 5 signal check enabled
Local> define port 5 signal control disabled speed 9600
Local> define port 5 type hardcopy
Local> logout port 5</td></tr>
<tr><td>4</td><td>Configure the TELNET server characteristics of the terminal server port for the printer. For example, to set up TELNET server carriage control handling:

Local> define port 5 telnet server newline to host <CRLF>
Local> set port 5 telnet server newline to host <CRLF></td></tr>
<tr><td>5</td><td>Configure the TELNET listener port to associate the listener with the printer port. The valid TCP listener port numbers for the DECserver 300 are 2001 through 2016. For example:

Local> define telnet listener 2005 ports 5 enabled
Local> set telnet listener 2005 ports 5 enabled
Local> define telnet listener 2005 identification "PR1"
Local> set telnet listener 2005 identification "PR1"
Local> define telnet listener 2005 connection enabled
Local> set telnet listener 2005 connection enabled</td></tr>
</table>

6	On the OpenVMS host, initialize and start up the print queue, as follows:
	`$ INIT/QUEUE/PROCESS=TCPWARE_TSSYM/ON="192.168.95.100,2005" PR1`
	`$ START/QUEUE PR1`

OpenVMS users can now issue print commands to the printer, such as:

```
$ PRINT/HEADER/QUEUE=PR1/COPIES=10 TEST1, TEST2, TEST3
```

TSSYM Tuning Logicals

TSSYM provides the retry interval and timeout tuning logicals (all are executive mode system logicals) listed in Table 14-9. See *LPS System Logicals*.

Table 14-9   TSSYM Tuning Logicals

TSSYM logical...	Description
TCPWARE_TSSYM_*_ RETRY_INTERVAL	Defines the interval at which the symbiont retries to make a connection to a printer after an attempt fails. The default is 0::15 (15 seconds delta time).
TCPWARE_TSSYM_qname_ RETRY_INTERVAL	Same as TCPWARE_TSSYM_*_RETRY_INTERVAL, but for a specific queue only, and overrides TCPWARE_TSSYM_*_RETRY_INTERVAL.
TCPWARE_TSSYM_*_ TIMEOUT	Defines the time it takes for a print job to abort if the connection to the printer is never established. The default timeout is infinite (it never times out).
TCPWARE_TSSYM_qname_ TIMEOUT	Same as TCPWARE_TSSYM_*_TIMEOUT, but for a specific queue only, and overrides TCPWARE_TSSYM_*_TIMEOUT.

Troubleshooting TSSYM

OPCOM can send a number of status messages that can help you troubleshoot TSSYM.

Access error messages help by entering **`HELP TCPWARE MESSAGES`**.

Chapter 15

Managing R Commands

Introduction

This chapter describes how to:

- Manage and troubleshoot the Berkeley R Commands Services, and set up host equivalence files on the server.
- Manage the R Shell (RSH) server, which handles Remote Copy Program (RCP) requests.
- Manage the Remote Magnetic Tape (RMT) Server.

R Services

Configure the Berkeley R Services as part of TCPware configuration (CNFNET). The following are the specific steps in the process:

1	Specify whether you want to enable the `login`, `shell`, and `exec` services. Specify YES or NO in each case.
2	For `login` service, also specify whether you want NORMAL or SECURE login authorization (the default is SECURE), as described in Table 15-2.
3	If desired, create a service access list (as described in the *Service Access Lists* section).
4	Set up a host equivalence file (as described in the *Host Equivalence Files* section on the following page).

Service Access Lists

You may want to set up a service access list to control which hosts, group of hosts, or network can access the service.

Use the ADD ACCESS_LIST command in TCPware's Network Control Utility (NETCU) that lets you specify a list number and PERMIT or DENY condition for a specific network internet address (and optional network mask). Here is the command format:

```
$ NETCU ADD ACCESS_LIST list condition ia mask
```

For details on the ADD ACCESS_LIST command, see the *NETCU Command Reference*.

Host Equivalence Files

Host equivalence files are security access files on the server host used to authorize access to services by other hosts or users. The files list hostnames (and, optionally, usernames) and indicate which remote hosts and users have equivalent access as local users.

(Host equivalence files authorize access to `login` and `shell` services only, since the `exec` service relies on direct username and password authorization.)

Two types of equivalence files are available: `HOSTS.EQUIV` and `.RHOSTS`. These files serve slightly different needs and are in different locations on the OpenVMS host, although they use the same data format:

- The `HOSTS.EQUIV` file defines which remote hosts or users can have equivalent access to the server host, and is analogous to the `/etc/hosts.equiv` file in UNIX. Place the `HOSTS.EQUIV` file in either the TCPWARE_COMMON:[TCPWARE] or TCPWARE_SPECIFIC:[TCPWARE] directory, depending on your configuration.
- The `.RHOSTS` file lets remote users access local accounts beyond what the `HOSTS.EQUIV` file specifies, and is analogous to the UNIX `~/.rhosts` file. Place the `.RHOSTS` file in the local accounts login directory. Here are some things to keep in mind:
  - A remote user specified in a `.RHOSTS` file can access the local account only if the account owns the file.
  - Access to the SYSTEM (root) account, or one that has a system UIC group, requires a `.RHOSTS` file, and does not work in the `HOSTS.EQUIV` file.
  - To disable user-specified `SYS$LOGIN:.RHOSTS` files (and use the `HOSTS.EQUIV` file only), set the TCPWARE_RCMD_FLAGS system logical to **`1`** (it is **`0`** by default).

Both the `HOSTS.EQUIV` and `.RHOSTS` files contain line entries for hostnames and optional usernames, in the following format, where *host* is the name of the remote host allowed access and the optional *username* is the name of a specific user on that host:

host [username]

Here are some sample lines in a host equivalence file, with the following considerations:

```
ALPHA
BETA smith
GAMMA Jones
DELTA +
+ JackSprat
```

- If you specify only *host* on a line (omitting *username*), users having accounts with the same names on both hosts can access the local system.

- If you specify both *host* and *username* on a line, the specified user on that host can access the local system or account. The *username* must match case exactly with the incoming username (`JackSprat` does not match with `jacksprat`).
- You can use the plus symbol (**+**) as a wildcard for *host* and *username*, but do not use an asterisk (*****) since UNIX does not recognize it as a wildcard. A wildcarded *host* entry grants access to any host, provided username checks pass. A wilcarded *username* entry grants access to any user, provided the hostname passes.
- The `HOSTS.EQUIV` and `.RHOSTS` files may handle some entries differently, as described in Table 15-1.

Table 15-1 Usernames and Wildcards in `HOSTS.EQUIV` and `.RHOSTS` Files

Sample entry...	In the `HOSTS.EQUIV` file...	In the `.RHOSTS` file (with account owner Betty)...
`ALPHA`	Accepts anyone from ALPHA for access to a local account with the same name as the remote username	Accepts only username Betty from ALPHA
`ALPHA Betty`	Accepts only username Betty from ALPHA for any account other than SYSTEM	Accepts only username Betty from ALPHA
`ALPHA David`	Accepts only username David from ALPHA for any account other than SYSTEM	Accepts only username David from ALPHA who requests username Betty (such as with `rlogin /user="Betty"`)
`ALPHA +`	Accepts anyone from ALPHA for any requested account	Accepts anyone from ALPHA who requests username Betty
`+ Betty`	Accepts only username Betty from any host for any account other than SYSTEM	Accepts only username Betty from any host
`+ David`	Accepts only username David from any host for any account other than SYSTEM	Accepts only username David from any host who requests username Betty
`+ +`	Accepts anyone from any host for any requested account other than SYSTEM	Accepts anyone from any host who requests username Betty

CAUTION! Use *username* values or wildcards in the `HOSTS.EQUIV` file with security in mind. Because `HOSTS.EQUIV` files apply system-wide, any remote user (or group of users) can masquerade as a local user. Especially avoid using double pluses (**`+ +`**) in either file.

Host equivalence files grant access authorization differently depending on how you configure each

R Service during the regular TCPware configuration:

- For `login` service, CNFNET offers NORMAL and SECURE options for login authorization (see Table 15-2).

Table 15-2 `login` Service Access Matrix

When...	With NORMAL `login...`	With SECURE `login...`
Hostname and username appear in the `.RHOSTS` or `HOSTS.EQUIV` file	User is logged in	User is prompted for password
Hostname or username does not appear in the `.RHOSTS` or `HOSTS.EQUIV` file	User is prompted for username and password (standard login sequence)	Access is denied

- The `shell` service checks the host equivalence files for command execution. If the check fails, `shell` denies access.
- The `exec` service performs a standard username and password check before allowing command execution on the host.

  The `exec` service also tracks break-in attempts using the OpenVMS Intrusion database. If a remote user enters an invalid login, TCPware creates an intrusion record based on the remote source's IP address. If TCPware reaches the threshold number of invalid login attempts, no one at the remote IP address can use the `exec` service. The Intrusion database shows who the offending addresses are; you can re-enable login attempts by deleting the intrusion records.

See the SHOW INTRUSION and DELETE/INTRUSION_RECORD commands in Compaq's *VMS DCL Dictionary* for details.

Customizing the `shell` and exec Services

Incoming `shell` and `exec` services invoke the TCPWARE:TCPWARE_RSERVICE.COM procedure to perform the requested operation. You can customize this command procedure to map an incoming command to an OpenVMS command. (Just make sure that you do not destroy mapping for TCPware operations such as RCP and RMT.)

The REXEC and the RSHELL servers have the functionality to set up a DECwindows display to the client's IP address if the TCP/IP transport is loaded. To enable this functionality, define the logical as:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_RCMD_ENABLE_DISPLAY “TRUE“
```

R Services Log File

You can set up a log file for incoming R Services such as RCP and RSH by defining the TCPWARE_RCMD_OUTPUT logical to log messages in the RCMD.LOG file, as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_RCMD_OUTPUT RCMD.LOG
```

Troubleshooting R Services

Access error messages help by entering **HELP TCPWARE MESSAGES** ***[identifier]***, or connect to web site **http://www.process.com** (select **Customer Support** followed by the **Error Messages** button).

RCP Server

Use the SHOW SERVICES command in NETCU to make sure that the R shell daemon (`rshd`) is operating before client users can use the RCP command. On TCPware hosts, the service should show up as `shell` under the "Port" column. If it does not, use CNFNET to enable the R Services.

You also need to use the `rexec` server if client users need to specify the RCP command's /USER, /PASSWORD, and /TRUNCATE qualifiers. On TCPware hosts, the service shows up as `exec` under the "Port" column.

CAUTION! The /PASSWORD qualifier requires entry of a plain text password, which could cause a security problem. You can avoid having users specify the /USER or /PASSWORD qualifier by checking that remote hosts include your hostname entry in their host equivalence files (such as the `/etc/hosts.equiv` file in UNIX systems, or the `TCPWARE:HOSTS.EQUIV` or `SYS$LOGIN:.RHOSTS` file locally).

Make sure that your users' login files (as well as the `~/.login` files on UNIX system clients) do not contain commands that generate output.

You can log RCP activity by defining the TCPWARE_RCMD_OUTPUT logical. (See *R Services Log File*.)

Troubleshooting RCP

Here are some steps to follow to ensure that the RCP command given from a UNIX system works correctly with TCPware's R Services. First make sure that the R Services were configured by running @TCPWARE:CNFNET RCMD to make sure that RCP is at the very least enabled and the image installed. Then follow these steps:

<u>On the TCPware system</u>:

1	Set the default to the login directory of the user account to which you will be copying files.
2	Edit the `SYS$LOGIN:.RHOSTS` file and add the UNIX hostname and username.
3	Edit the `TCPWARE:HOSTS.` file and add the IP address and hostname of the UNIX system from which you will be copying files.
4	Check the user account's UAF record (using SET DEFAULT SYS$SYSTEM followed by MCR AUTHORIZE).

5	Check that the owner of the login directory and `.RHOSTS` file is the same as the UAF record's IDENTIFIER/UIC (using DIRECTORY/OWNER). If different, you must set the ownership of the login directory and .RHOSTS file to match the IDENTIFIER/UIC.

On the UNIX system:

1	Issue the `rcp` command. If you get a `Login information not recognized at remote node` message, run NETCU DEBUG /TCP /DATA=1500 /DIA=*unix-host-ip-address*, preferably with the additional /OUTPUT=RCPDEBUG.LOG.
2	If there is a debug trace, look for the *`unix.username.vax-username`*`.rcp` *`filename.ext`*. If you notice that the case of the username is different from how it appears in the `.RHOSTS` file, change the case in the `.RHOSTS` file to match.

A `permission denied` message usually indicates a protection error on the UNIX side.

RMT Server

Here are some preliminary tasks to make sure the remote client can access the tape or CD-ROM drive:

1	Make sure the `shell` and `exec` services and the RMTSETUP image are installed during TCPware configuration at the `Do you want to activate ... service:` and `Do you want to INSTALL the ... image:` prompts, or make sure that at least `RCMD_SERVICES == ":SHELL:EXEC"` and `RCMD_CLIENTS == ":RMTSETUP"` appear in your TCPWARE:TCPWARE_CONFIGURE.COM file.
2	Make sure the magnetic tape or CD-ROM, recognizable to the system, is loaded in the device. With a tape device, the client essentially 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 issuing a MOUNT command.
3	Make sure the `rsh` command works from the UNIX system user's `root` directory to the OpenVMS user SYSTEM's directory.
4	For SYSTEM accounts, RMT ignores the `TCPWARE:HOSTS.EQUIV` file and uses an explicit entry in the `SYS$LOGIN:.RHOSTS` file to grant access.
5	Make sure nothing is output from SYS$SYLOGIN procedure or from SYS$LOGIN:LOGIN.COM when issuing the `rsh` command to SYSTEM.

6	Make sure you suppress output by including the following commands in the SYS$SYLOGIN and LOGIN.COM procedures:
	`$ RMT_VERIFY = 'F$VERIFY(0)`
	`.`
	`.`
	`.`
	`$ IF (F$MODE() .NES. "OTHER") THEN RMT_VERIFY = F$VERIFY(RMT_VERIFY)`

RMT Client Utilities

On the remote host, a user can employ the `rdump` utility to dump files to OpenVMS tapes, or the `rrestore` utility 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 TCPware's RMT service. For example, not all UNIX systems let the user restore files selectively using `rrestore`.

When remote users use these remote dump and restore commands, they must specify either the OpenVMS device name or a filename. If they specify a device name, it must be a valid OpenVMS type of magnetic tape device name.

See the *SunOS Reference Manual*, the *Maintenance* chapter, the sections on `dump`, `rdump`, `restore`, and `rrestore`, or your particular client system's documentation for details. Users should be careful about the order in which they specify options on the command line.

Here is an example of an `rdump` command:

```
> /etc/rdump 0f lilac:mua0:/nomount /usr
```

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.

TCPware's RMT Server lets you specify the qualifiers with magnetic tape device names indicated in Table 15-3. The qualifiers are available with the RMTSETUP command.

Table 15-3 RMT Magtape Qualifiers

Qualifier...	**Defines...**
/ASSIST /NOASSIST	Whether to use operator assistance to mount the volume. /ASSIST is the default.
/BLOCKSIZE=*n*	Default block size for magnetic tape volumes. The default is 65534 bytes.
/COMMENT="*string*"	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.
/DENSITY=*n*	Density (in bits per inch) at which to write a foreign or unlabeled magnetic tape. The default is the current density.

Table 15-3 RMT Magtape Qualifiers (Continued)

Qualifier...	Defines...
/MOUNT /NOMOUNT	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. With /NOMOUNT, the qualifiers /ASSIST, /BLOCKSIZE, /COMMENT, and /DENSITY are not allowed. With /NOMOUNT, the defaults are /NOUNLOAD, /NOREWIND, /STREAM, and /WRITE.
/REWIND /NOREWIND	Whether to rewind the drive when it is closed. The default is /REWIND.
/STREAM /NOSTREAM	Whether to read the tape in record (/NOSTREAM) or byte-stream (/STREAM) mode. The default is /STREAM.
/UNLOAD /NOUNLOAD	Whether to unload the drive when it is closed. The default is /UNLOAD.
/WRITE /NOWRITE	Whether you can write to the volume. The default is /WRITE.

Table 15-4 indicates the supported option for remote access to the CD-ROM drive.

Table 15-4 RMT CD-ROM Qualifiers

Qualifier...	Defines...
/CD	Indicates that the remote device is a CD-ROM device.

Client Examples

The following steps perform `rdump` and `rrestore` from a Sun UNIX client system. They dump two Sun directories to the tape by issuing separate `rdump` commands. They then restore files selectively from the tape to the Sun client system:

1 Put the directories on the tape by issuing two **`rdump`** commands:

```
SUN> /etc/rdump 0f homer:mkb600/nomount /
SUN> /etc/rdump 0f homer:mkb600/nomount/rewind /usr
```

Include the `/nomount` qualifier with the first `rdump` to keep OpenVMS from rewinding the tape (even though `rdump` on the Sun reports to the contrary). The additional `/rewind` qualifier for the second `rdump` actually rewinds the tape.

2	Restore the files selectively from the tape using **`rrestore`**. In this example, `rrestore` extracts `.rlogin` from the second (**`2`**) dump file on the tape: `SUN>` **`/etc/rrestore fsx homer:mkb600/nomount/rewind 2 .rlogin`** In this example, `rrestore` invokes the interactive utility to let the user specify particular files that were put on the tape in the first (**`1`**) dump file. The `add` command then adds the files to the extraction list and the `extract` command restores them: `SUN>` **`/etc/rrestore fis homer:mkb600/unload 1`** `restore>` **`add /users`** `restore>` **`extract`** The `rrestore` command may display messages such as `You have not read any volumes yet` and ask you to specify the next volume. Although the messages may appear, `rrestore` should work properly.

Chapter 16

Managing Mail Services

This chapter describes how to configure the TCPware SMTP (Simple Mail Transport Protocol) server to send and receive electronic mail.

The chapter also includes information about the Internet Message Access Protocol (IMAP) and the Post Office Protocol (POP).

If you are running PMDF or another mail system that provides its own SMTP support, refer to that mail system's documentation.

- For information about modifying the TCPware SMTP configuration file, see the *Modifying the TCPware SMTP Configuration File* section.
- For information about configuring mail parameters, see the *Configuring Mail Parameters* section.
- For information about configuring the SMTP server for inbound mail, see the *Configuring the SMTP Server for Inbound Mail* section.
- For information about configuring the SMTP symbiont and mail queues for outbound mail, see the *Configuring the SMTP Symbiont and Mail Queues for Outbound Mail* section.
- For information about using the IMAP Server for accessing remote message storage, see the *IMAP Server* section.
- For information about delivering inbound mail to remote hosts via Post Office Protocol (POP), see the *Post Office Protocol (POP) Versions 2 and 3* section.
- For information about configuring SMTP service for ALL-IN-1 users, see the *Configuring SMTP Service for ALL-IN-1 Users* section.
- For information about transferring mail between TCPware and DECnet-only hosts, see the *Configuring the SMTP-DECnet Mail Gateway* section.

Modifying the TCPware SMTP Configuration File

TCPware SMTP configuration is stored in the START_SMTP.COM and START_SMTP_LOCAL.COM startup command procedures. TCPware provides a utility for editing these files:

MAIL-CONFIG	Invoked with the TCPWARE CONFIGURE /MAIL command

After using these configuration utilities, stop and restart the mail queues with @TCPWARE:START_SMTP.COM to update the VMScluster or with @TCPWARE:START_SMTP_LOCAL.COM to update the local host only.

Pipelining and Extended SMTP

The current release of SMTP implements Extended SMTP (RFC-1869) and Pipelining (RFC-2197)

Delivering Mail to Specific Folders

The SMTP server supports mail delivery to folders other than the NEWMAIL folder. The foldernames are restricted to UPPERCASE characters only, the pound sign (#), and the underscore (_). Use of the comma (,) in a foldername causes an error. Mail addressed to *user+folder@host* is delivered to the specified *folder*. You can disable this mechanism by defining the system-wide logical name TCPWARE_SMTP_DISABLE_FOLDER_DELIVERY.

Using the New Mail Delivery Mechanisms

The current release of SMTP supports alias file extensions that request mail delivery to a file or specify addresses in a separate file. You must use the SMTP aliases file, specified with TCPWARE CONFIG/MAIL, to list all of these new mail delivery mechanisms. The default is TCPWARE:SMTP_ALIASES. The syntax for these aliases follows the form of those described in *Configuring Mail Aliases* found later in this chapter. It is necessary to use the colon and semicolon in the command lines as shown in the examples.

<table>
<tr><td><device:[directory]address.list</td><td>Delivers mail to the list of addresses in the specified file.

<code>alias1 : "<filespec" ;</code></td></tr>
<tr><td>| device:[directory]procedure.com parameter(s)</td><td>Submits the specified command procedure to the queue identified by the logical name TCPWARE_SMTP_BATCH_QUEUE, SYS$BATCH by default. The first parameter (P1), passed to the submitted procedure, is always the name of a temporary file containing the mail message that the procedure must delete. Any parameter(s) specified in the alias file are passed to the submitted procedure in a single string as its second parameter (P2).

<code>alias2 : "|filespec p2 p3" ;</code></td></tr>
</table>

>*device:[directory]mail.file*	Appends mail to the specified file. If no filetype is specified, the default is *.yyyy-mm*, where: • *yyyy* and *mm* are to the current year and month, respectively. `alias3 : ">filespec" ;`

Rejecting Mail Messages

The SMTP server supports a set of rules for rejecting mail messages received by itself based on the mail header contents or any combination of MAIL FROM, RCPT TO, and Source IP Address values. Mail matching the criteria can be quietly ignored or rejected with a message to the SMTP client or delivered to an address rewritten according to the rule specification. This capability can be useful for controlling SPAM and preventing your system from being used as a mail relay.

The file TCPWARE:SMTP_SERVER_REJECT. contains the rejection and rewrite rules. You may specify an alternate file via the logical name TCPWARE_SMTP_SERVER_REJECT_FILE. A rejection file line of the form **#include** *device:[directory]reject.file* temporarily suspends processing of the current file and begins processing of the specified file. Rejection files can be nested to arbitrary depth. Comments may be included in rejection files by placing any of the characters ; or ! or # in the first column of a line.

The following is a sample rejection file:

```
!
! This is a sample reject file for the company FLOWERS.COM.
!
! Entries can have one of the following formats:
!
!    from_user [from_ip to_user action action-data]
!
!    :rfc822 header
!
! Wildcards can be used in FROM_USER, FROM_IP, and TO_USER. ACTION is the
! reject action, which is one of:
!
!   n     Don't reject, but rewrite TO address to be ACTION-DATA.
!         If ACTION-DATA is blank then we simply deliver to TO_USER.
!
!   y     Reject and use optional ACTION-FIELD as a rejection message
!         format that can contain up to three %s formatting designators
!         for mail from, mail to, and local domainname.
!
!   q     Reject quietly -- don't inform Sending SMTP Client that
!         message will be discarded. If only FROM_USER is specified
!         other fields default to FROM_IP=*, TO_USER=*, and ACTION=n.
!
! Accept all messages with MAIL FROM:<> (bounce messages)
!
```

```
<>   *   *   n
!
! Don't rewrite or reject any mail to "postmaster*"
!
*    *    postmaster*n
!
! Reject anything with a Message-ID that appears to have originated from
! cyberpromo.com or nowhere.com
!
Message-ID: <*@cyberpromo.com>
Message-ID: <*@nowhere.com>
!
! Reject mail from well-known SPAM sites with sample non-standard error
! messages.
!
<*answerme.com>    *  *  y  "Spam from <%s> rejected"
<*cyberpromo.com>  *  *  y  "Spam from <%s> to <%s> rejected"
<*pleaseread.com>  *  *  y  "Spam rejected;%.0s%.0s Contact postmaster@%s"
!
!  Disallow percent-hacks (e.g, joe%somewhere.com@flowers.com)
!
*  *  *@*@*flowers.com  y   "No forwarding-path relaying allowed"
!
!  Rewrite all mail to webmaster to the postmaster
!
*  *  webmaster@*flowers.com  n  postmaster@flowers.com
!
!
! Disallow relaying through our mailer, and only allow users on our
! networks to claim to be from our company (flowers.com)
!
*    *  *@flowers.com  n
*    *  *@daisy.flowers.com  n
*    *  *@[10.0.0.1]   n
!
<*flowers.com>   10.0.0.*   *   n
<*flowers.com>   10.115.140.*   *  n
<*flowers.com>   10.115.141.*   *  n
!
*@*   *  *@*    y "no relaying through this site"
*     *  *@*    y "missing domain name in MAIL PATH"
!
!end of sample file
```

Mail rejection rules have two formats:

- *:RFC822_header pattern*

  This format causes rejection of any mail in which a line with the specified header matches the given *pattern*. The following rejection message is sent to the client:

  ```
  554 Message rejected due to header contents
  ```

Note! Use caution when rejecting mail based on header contents. No other criteria are considered during rejection processing.

- from_user ip_address to_user action action_data

  This format causes rejection or alternate delivery of all messages that match all of the patterns specified. The *action* item can be as follows:

n	Means do not reject the mail, but deliver it to the address specified as the *action_data*. If *action_data* is not specified, deliver the message to its intended recipient.

y	Means reject the mail, sending the *action_data* string to the SMTP client as a rejection message. The *action_data* item is actually used as a format string and may contain from one to three %s formatting designators to include the *from_user*, the *to_user*, and the SMTP server name, **in that order**. If *action_data* is missing, the default rejection message is
	
	`553-Mail to <to_user> not allowed;`
	`553 Contact Postmaster@<smtpserver> to remove block`

q	Means reject the mail, but do not give the SMTP client any indication that it has been rejected. Use caution when rejecting messages quietly.

Each of the pattern specifications *pattern*, *from_user*, *ip_address*, and *to_user* may contain the OpenVMS * and % wildcard characters.

You can represent `from_addr` expressions in the SMTP_SERVER_REJECT filter with the <> syntax. So, `*@*domain.com` and `<*@*domain.com>` are the same expression. To allow all bounce mail in without the filter rules being applied, add the following line to the top of your SMTP_SERVER_REJECT file:

```
<> * * n
(Accept any mail with a MAIL FROM: of <>)
```

When comparing the RCPT TO: address with the SMTP_SERVER_REJECT file expressions, any '%' signs in the RCPT TO: address are changed to '@'. You can write filter rules in the SMTP_SERVER_REJECT files that can match against forward-path relays. You can add the rule of

```
* * *@*@localdomain y "No forward-path relaying allowed"
```

to your SMTP_SERVER_REJECT file above the rules that accept mail with the destination of your domains. RCPT TO.: addresses will replace any % character with the @ character for matching purposes only so you can filter with *@*@*-type rules. So, RCPT TO.:<xxx%yyy@zzz> is changed to xxx@yyy@zzz. You can use the logical name TCPWARE_SMTP_SERVER_REJECT_INFO to control debug and informational OPCOM messages produced during rejection processing. You should define it to have some non-zero value to request OPCOM messages. The following values may be combined to control message quantity and content:

Values	To show...
1	mail rejected due to *action* **y**
2	rewritten addresses (*action* **n** with *action_data*)
4	the reject message sent to the remote system
8	configuration file parsing
16	non-written addresses (*action* **n** and no *action_data*)
32	mail rejected due to *action* **q**
64	mail rejected due to header rules

The value 65 is appropriate for auditing rejection activity.

The remainder of this chapter describes the configuration tasks.

For general information on the MAIL-CONFIG utility, see Chapter 2.

Configuring Mail Parameters

The parameters that control the operations of the TCPware mailer are described in Table 16-1.

Configuring Mail Parameters with MAIL-CONFIG

To configure mail parameters with the MAIL-CONFIG utility:

1	Start MAIL-CONFIG with the TCPWARE CONFIGURE /MAIL command.
2	Use the SET *parameter_name* commands.
3	Save the configuration with the SAVE command.
4	Quit MAIL-CONFIG with the QUIT command.

The modified configuration takes effect the next time your system reboots or the queues are restarted.

Mail Parameters

Table 16-1 describes all the mail parameters you can set with the MAIL-CONFIG utility.

Table 16-1 Mail Parameters

Parameter	Description
ALIAS-FILE	File in which SMTP aliases are stored; see the *Configuring Mail Aliases* section.
DECNET-DOMAIN	Domain name for DECnet gateway function; see the *Configuring the SMTP-DECnet Mail Gateway* section.
DELIVERY-RECEIPTS	When TRUE, causes the SMTP processor to honor Delivery-Receipt-To: headers.
DISABLE-PSIMAIL	When TRUE, the SMTP symbiont looks for messages addressed through PSImail (usually of the form PSI%*address*::*user*) and returns messages with those addresses to the sender marked "user unknown."
DISALLOW-USER-REPLY-TO	When TRUE, prevents VMS MAIL users from setting a Reply-To: header address with the logical name TCPWARE_SMTP_REPLY_TO.
FORWARDER	Identifies a host (known as a *mail hub*) to which mail should be forwarded if a host name cannot be resolved for an address. The specified name must resolve to an IP address; it must not resolve to an MX record.
FORWARD-LOCAL-MAIL	Forwards all mail designated for local users to the mail hub instead of delivering it locally. Can be overridden by entries in the SMTP_ALIASES file.
FORWARD-REMOTE-MAIL	Forwards all SMTP-delivered mail to the mail hub instead of directly to the destination host. Can be overridden by a GATEWAY or LOCAL-DOMAIN entry.
HEADER-CONTROL	Controls which RFC-822 headers appear in messages delivered to VMS MAIL users.
HOST-ALIAS-FILE	Contains a list of host names considered aliases for the local host name.
LOCAL-MAIL-FORWARDER	Identifies a host to which mail should be forwarded when a local mail delivery fails because the user name is unknown.
POSTMASTER	Identifies the user name of the system postmaster.
QUEUE-COUNT	Specifies the number of mail processing queues to create on a particular system.

Table 16-1 Mail Parameters (Continued)

Parameter	Description
REPLY-CONTROL	Specifies how Internet mail headers should be mapped to the VMS MAIL from header. Permitted values are FROM and REPLY-TO. You may specify both as a comma-separated list.
RESENT-HEADERS	When FALSE, the SMTP symbiont omits the Resent-From, Resent-To, and Resent-Date headers that are usually included when a message is forwarded using a VMS MAIL forwarding address.
RETRY-INTERVAL	Specifies the amount of time (in minutes) that should elapse after a failed attempt before another attempt is made.
RETURN-INTERVAL	Specifies the amount of time (in hours) a message can remain in the processing queue before it is returned to sender.
SEND-BROADCAST-CLASS	Controls the OpenVMS broadcast class used by SMTP for SEND-type messages (which are sent to a terminal).
SMTP-HOST-NAMES	A list of up to 16 host names to consider as aliases for the local host. See the *Specifying SMTP Host Aliases* section.
START-QUEUE-MANAGER	Determines whether START_SMTP.COM starts the VMS queue manager if it is not already running.

Configuring the SMTP Server for Inbound Mail

The TCPware SMTP server accepts mail from remote hosts and delivers it to users' mailboxes.

By default, the SMTP server is enabled when you install TCPware. For details on configuring and controlling TCPware servers, see Chapter 4.

Translating UNIX-Style Linefeeds to SMTP-Compliant End-of-Line Character Sequences

TCPware provides a logical name to solve the problem of systems sending messages containing lines terminated by an LF character only instead of the proper CR/LF sequence. The following command tells TCPware to accept the bare LF as the end-of-line indicator:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_SMTP_ACCEPT_UNIX_LF_BRAIN_DAMAGE TRUE
```

TCPware lets you validate the contents of the envelope-from field by defining the system-wide logical name TCPWARE_SMTP_REJECT_INVALID_DOMAINS. Use the equivalence string STRICT to require the presence of a host in those addresses. For example, require MAIL FROM: <*user@host*> rather than MAIL FROM: <*user*>. The host specified in the MAIL FROM: address must exist in the DNS database.

Configuring the SMTP Symbiont and Mail Queues for Outbound Mail

TCPware lets users send mail to remote destinations by submitting outbound messages to mail queues that are processed by TCPware's SMTP symbiont.

You can configure the SMTP symbiont to:

- Control users' ability to specify their own REPLY-To: headers (see the *Specifying the REPLY_TO Header* section.)
- Provide more than one server queue for each cluster node. By default, TCPware provides one server queue for each cluster node running TCPware (see the *Configuring Mail Queues* section.).
- Forward mail through a central mail hub (see the *Forwarding Mail through a Mail Hub* section.)
- Use gateways to reach specific hosts, domains, or "virtual" domains (see the *Configuring Mail Gateways* section.)
- Use host aliases (see the *Specifying SMTP Host Aliases* section.)
- Use mail aliases (see the *Configuring Mail Aliases* section.)

You can also write your own SMTP dispatcher by modifying and compiling the SMTP user exit TCPWARE_ROOT:[TCPWARE.EXAMPLES]USER_SMTP_DISPATCH.C. Instructions for modifying the dispatcher are outside the scope of this manual.

For outbound mail, TCPware SMTP eases the 255 character limitation on RFC-822 To: and CC: header lengths. The limit of 255 characters was imposed because some mail applications cannot handle headers longer than 255 characters.

The default header length is 1024 characters. The maximum length is configurable by defining the logical name TCPWARE_SMTP_MAXIMUM_822_TO_LENGTH to be the maximum number of characters to allow in the header. If that maximum is exceeded, only as many addresses as will fit are put into the header. VMSmail always creates X-VMSmail-To: and -CC: headers that contain all of the given addresses.

To automatically lower the case of usernames in outbound messages, define the logical name TCPWARE_VMSMAIL_LOCASE_USERNAME.

Specifying the REPLY_TO Header

The TCPWARE_SMTP_REPLY_TO logical name lets you specify the value for the RFC822 REPLY-TO: header. For example, to set your Reply-To: header to FNORD@FLOWERS.COM, use the command:

```
$ DEFINE TCPWARE_SMTP_REPLY_TO "FNORD@FLOWERS.COM"
```

This logical name only affects mail agents that use the SMTP% interface (for example, OpenVMS and DECwindows mail). The system manager can disable the use of this logical name with the SET DISALLOW-USER-REPLY-TO command of the TCPWARE CONFIGURE /MAIL utility.

Disabling VRFY and EXPN

To disable VRFY and EXPN processing, use the logical name TCPWARE_SMTP_SERVER_DISABLE_VRFYEXPN. Define it to have some non-zero value to disable the requisite functions. The following values may be combined to specify which function:

Value	Function
1	to disable VRFY
2	to disable EXPN
3	to disable both VRFY and EXPN

Configuring Mail Queues

TCPware uses OpenVMS server queues for SMTP processing. Initially, TCPware configures each cluster node running TCPware with a server queue and configures a generic queue for the entire cluster. New messages are placed in the generic queue for processing, which distributes mail processing to the first available server queue.

For example, if three clustered nodes, Huey, Louey, and Dewey, are running TCPware, TCPware creates three server queues and one generic queue. The queue names are:

```
SMTP_HUEY        [Execution queue]
SMTP_LOUEY       [Execution queue]
SMTP_DEWEY       [Execution queue]
TCPWARE_SMTP     [Generic queue]
```

The following example lists the queues for node Huey:

```
$ SHOW QUEUE TCPWARE_SMTP/FULL
Generic server queue TCPWARE_SMTP
       /GENERIC=(SMTP_HUEY,SMTP_LOUEY,SMTP_DEWEY) /OWNER=[SYSTEM]
       /PROTECTION=(codes)


$ SHOW QUEUE SMTP_HUEY/FULL
Server queue SMTP_HUEY, idle, on HUEY::, mounted form DEFAULT
     /BASE_PRIORITY=4 /DEFAULT=(FEED,FORM=DEFAULT) /OWNER=[SYSTEM]
     /PROCESSOR=TCPWARE_SMTP_SYMBIONT /PROTECTION=(codes)
```

The queues SMTP_LOUEY and SMTP_DEWEY are also created, and are similar to the SMTP_HUEY queue shown. Note that a standalone (non-clustered machine) has two queues created by default; that is, one generic queue (TCPWARE_SMTP) and one execution queue (SMTP_ *nodename*).

Configuring Multiple Queues

If mail traffic is heavy on your system, you can configure multiple server queues on one or more nodes using MAIL-CONFIG.

To configure multiple queues with the MAIL-CONFIG utility:

1	Start MAIL-CONFIG with the TCPWARE CONFIGURE /MAIL command.
2	Use the SET QUEUE-COUNT command to specify the number of queues on the node.
3	Save the configuration with the SAVE command.
4	Quit MAIL-CONFIG with the QUIT command.

The modified configuration takes effect the next time your system reboots.

Configuring Queue Groups

In heterogeneous cluster environments, you may need to partition mail processing by grouping homogeneous subsets of your cluster into queue groups using MAIL-CONFIG.

To configure queue groups with the MAIL-CONFIG utility:

1	Start MAIL-CONFIG with the TCPWARE CONFIGURE /MAIL command.
2	Use the ADD QUEUE-GROUP and DELETE QUEUE-GROUP commands to add or delete queues.
3	Save the configuration with the SAVE command.
4	Quit MAIL-CONFIG with the QUIT command.

The modified configuration takes effect the next time your system reboots.

Forwarding Mail through a Mail Hub

Many sites provide outbound e-mail access to the Internet through a single system known as a *mail hub* to deliver all outbound mail on behalf of the other hosts at the site. A mail hub typically implements a single-address scheme for e-mail users at the site, so that all users have addresses of the form *username@sitename* rather than *username@hostname.sitename*. Site administrators often configure mail hubs to provide Internet e-mail access to hosts that do not have direct access to the Internet.

To forward mail through a mail hub:

1	Specify the host that will serve as a mail hub.
2	Specify the conditions under which TCPware forwards mail to the mail hub.

The following sections describe these procedures.

Specifying a Mail Hub

To specify the host that will serve as a mail hub for your TCPware host:

1	Start MAIL-CONFIG (TCPWARE CONFIGURE /MAIL).
2	Modify the FORWARDER parameter: • With MAIL-CONFIG, use the SET FORWARDER *mailhub_hostname* command.
3	If desired, set any of the following conditions for forwarding mail to the mail hub: • Forward mail addressed to users on remote hosts (see the *Forwarding Mail Addressed to Remote Hosts* section.) • Exclude hosts in specific domains from remote mail hub forwarding (see the *Excluding Hosts in Specific Domains From Mail Forwarding* section.) • Forward mail addressed to users on the local host (see the *Forwarding Local Mail* section.) Exclude specific local users from mail hub forwarding (see the *Excluding Specific Local Users from Mail Forwarding* section.)
4	Exit the configuration utility. When prompted, save the new parameters.
5	To make the changes take effect immediately, stop and restart the mail queues. To update the VMScluster, use the @TCPWARE:START_SMTP.COM command. To update the local host only, use the @TCPWARE:START_SMTP_LOCAL.COM command. Otherwise, your changes take effect the next time you reboot your system.

Forwarding Mail Addressed to Remote Hosts

To configure TCPware to forward mail addressed to remote users via a mail hub:

1	Make sure the FORWARDER parameter specifies the host you want to use as a mail hub (see the *Specifying a Mail Hub* section.).
2	Start MAIL-CONFIG (TCPWARE CONFIGURE /MAIL).
3	Modify the FORWARD-REMOTE-MAIL parameter: • With MAIL-CONFIG, use the SET FORWARD-REMOTE-MAIL TRUE command.
4	If desired, exclude hosts in specific domains from mail hub forwarding (see the *Excluding Hosts in Specific Domains from Mail Forwarding* section.).
5	If desired, specify other conditions under which TCPware forwards mail to the mail hub (see the *Specifying a Mail Hub* section.).
6	Exit the configuration utility. When prompted, save the new parameters.

7	To make the changes take effect immediately, stop and restart the mail queues with @TCPWARE:START_SMTP.COM to update the VMScluster or with @TCPWARE:START_SMTP_LOCAL.COM to update the local host only. Otherwise, your changes take effect the next time you reboot your system.

Excluding Hosts in Specific Domains From Mail Forwarding

If you configure TCPware to forward mail addressed to remote users via a mail hub (see the *Forwarding Local Mail* section), you can exclude hosts in specific domains from the mail forwarding system by adding the domain to a list of "local domains."

To modify the local domain list:

1	Make sure remote mail forwarding is enabled (see the *Forwarding Mail Addressed to Remote Hosts* section.).
2	Start MAIL-CONFIG (TCPWARE CONFIGURE /MAIL).
3	To add a domain to the list: • With MAIL-CONFIG, use the ADD LOCAL-DOMAIN *domain_name* command. If *domain_name* begins with a dot, it specifies a domain name. Otherwise, *domain_name* specifies a host name.
4	To delete a domain from the list: • With MAIL-CONFIG, use the DELETE LOCAL-DOMAIN *domain_name* command.
5	Exit the configuration utility. When prompted, save the modified configuration.
6	To make the new configuration take effect immediately, stop and restart the mail queues with @TCPWARE:START_SMTP.COM to update the VMScluster or with @TCPWARE:START_SMTP_LOCAL.COM to update the local host only. Otherwise, your changes take effect the next time you reboot your system.

Forwarding Local Mail

To configure TCPware to forward mail addressed to local users via a mail hub:

1	Make sure the FORWARDER parameter specifies the host you want to use as a mail hub (see the *Specifying a Mail Hub* section.).
2	Start MAIL-CONFIG (TCPWARE CONFIGURE /MAIL).
3	Modify the FORWARD-LOCAL-MAIL parameter: • With MAIL-CONFIG, use the SET FORWARD-LOCAL-MAIL TRUE command.
4	If desired, exclude specific local users from mail hub forwarding (see the *Excluding Specific Local Users from Mail Forwarding* section.).
5	If desired, specify other conditions under which TCPware forwards mail to the mail hub (see the *Specifying a Mail Hub* section.).
6	Exit the configuration utility. When prompted, save the new parameters.
7	To make the changes take effect immediately, stop and restart the mail queues with @TCPWARE:START_SMTP.COM to update the VMScluster or with @TCPWARE:START_SMTP_LOCAL.COM to update the local host only. Otherwise, your changes take effect the next time you restart your system.

Excluding Specific Local Users from Mail Forwarding

If you configure TCPware to forward local mail via a mail hub (see the *Forwarding Local Mail* section), you can exclude specific local users from the mail forwarding system by creating mail aliases for them in the TCPWARE:SMTP_ALIASES file. Each users' alias must be in the following format: `username: *;`

For more information on configuring mail aliases, see the *Configuring Mail Aliases* section.

Configuring Mail Gateways

You can configure TCPware with gateways to particular hosts or domains to override the normal host lookup used by SMTP or to configure "virtual" domains not actually present on the network. You can use MAIL-CONFIG.

To configure mail gateways with MAIL-CONFIG:

1	Start MAIL-CONFIG with the TCPWARE CONFIGURE /MAIL command.
2	Use the ADD GATEWAY and DELETE GATEWAY commands.
3	Save the configuration with the SAVE command.
4	Quit MAIL-CONFIG with the QUIT command.

The modified configuration takes effect the next time your system reboots.

Specifying SMTP Host Aliases

If your system is a member of a VMScluster, you can define *host aliases*, which are host names interpreted by the mailer as aliases for the actual local host name. You can specify these aliases in return addresses for individual users.

Setting Host Aliases

TCPware relies on two parameters to obtain its list of host aliases:

SMTP-HOST-NAMES	Is a comma-separated list of up to 16 host aliases. If defined, the first alias in the list is the name used for outgoing mail. Any aliases are names for which your host accepts incoming mail.
HOST-ALIAS-FILE	Is the complete file specification of a file containing an unlimited list of host alias entries (one entry per line). The HOST-ALIAS-FILE value defaults to TCPWARE:SMTP_HOST_ALIAS.

To change your host aliases with MAIL-CONFIG, use the SET SMTP-HOST-NAMES command or the SET HOST-ALIAS-FILE command and save the modified configuration with the SAVE command. The new configuration takes effect the next time you reboot the system or the queues are restarted.

Specifying Host Aliases for Individual Users

The logical name TCPWARE_SMTP_FROM_HOST lets you change the host name that appears in your return address on outgoing mail.

Normally, the host name you choose must be a "local" host name; that is, it must be one of the registered SMTP host name aliases on the system (either from the SMTP-HOST-NAMES setting or the HOST-ALIAS-FILE). If it is not a known alias, the setting is ignored.

If you define the host name in executive mode, however, TCPWARE_SMTP_FROM_HOST can be any arbitrary host name. The name is not checked against the SMTP host name.

This feature lets users from different administrative entities within an organization have return addresses that reflect the names of those entities. To enable this feature:

1	Set up MX records in DNS so mail is routed to the local host for each separate host name. For information about MX records, see the discussion of zone files in Chapter 6.
2	Set up SMTP-HOST-NAMES or the HOST-ALIAS-FILE with a list of host names.
3	Define the logical name TCPWARE_SMTP_FROM_HOST for each user. Base the value for this logical name on some aspect of the department or organization to which the user belongs.

Configuring Mail Aliases

The TCPware SMTP system supports system-wide mail aliases, system-wide mailing lists, and per-user mail aliases. The default system-wide alias file is TCPWARE:SMTP_ALIASES. You can configure this name or specify a list of alias file names.

Per-user mail aliases are kept in the file SMTP_ALIASES in each user's login directory. The format for alias entries is: `alias:  real_address[,...];`

– *alias* is an alphanumeric string.
– *real_address* is either a local or remote electronic mail address.

You can specify multiple addresses by separating them with commas; the alias definition may span multiple lines, if needed, and must always be terminated with a semicolon (;).

For example, a local user has the user name "JB134A", but wants to receive SMTP mail sent to the address "john". The system manager adds the following line to the alias file: `john:  jb134a;`

You can both forward a mail message and deliver it to a local mailbox by adding the mailbox name, preceded by an underscore, to the TCPWARE:SMTP_ALIASES file.

The following example shows such an alias entry:

```
FNORD: FNORD@SOMEWHERE.FLOWERS.COM, _FNORD;
```

The leading underscore on the second address (_FNORD), tells the SMTP symbiont to skip any further alias processing.

Mailing Lists

Mailing lists are a special form of mail alias and are supported only in the system-wide alias files. The format for specifying a mailing list is: `list-name:: owner-address, file-spec;`

where:

– A double-colon (::) signifies that this alias is a mailing list.
– *owner-address* is the address of the mailing list owner. Messages sent to this mailing list go to each subscriber on the list with the return-path set to this address. The owner address can be an actual user's address or an alias, if desired.
– *file-spec* is the file specification for the file containing the subscribers to the mailing list. Specify a complete path name for this file, including the device and directory.

For example, you might want to set up a mailing list called OPERATIONS-STAFF for your operations staff, and have your operations manager, user OPER1, manage that list. You might set up the mailing list this way:

```
Operations-Staff:: Operations-Manager, USERS:[OPER1]STAFF.LIST;
Operations-Manager: OPER1;
```

Mail sent to OPERATIONS-STAFF is forwarded to the addresses listed in USERS:[OPER1]STAFF.LIST. Because this file is in OPER1's area, the operations manager has control over who is included in the list. The list is set up in this example so the return-path on list messages is set to "Operations-Manager" instead of user OPER1; setting up the list owner as an alias makes it easier to change list owners at a later date.

Specifying the System-Wide Mail Alias File

By default, the TCPware SMTP system obtains system-wide mail aliases from the TCPWARE:SMTP_ALIASES file. You can configure TCPware to use any other file, or to use multiple files.

To change the SMTP aliases file with MAIL-CONFIG, use the SET ALIASES-FILE command, then save the modified configuration with the SAVE command. The new configuration takes effect the next time you reboot the system.

Using Mail Aliases and Mailing Lists From VMS MAIL

If you want aliases configured within the TCPware SMTP alias file to be accessible to local VMS MAIL users (or those connected via DECnet), specify the address using the TCPware SMTP VMS MAIL foreign mail protocol interface.

For example, a local user wishing to send mail to the "gcc-users" mailing list would specify the address SMTP%"gcc-users". Note that you can, however, define a VMS MAIL alias containing the SMTP% specification.

To define the VMS MAIL alias "Operations-Staff," use the VMS MAIL SET FORWARD command:

```
MAIL> SET FORWARD SMTP%"""Operations-Staff-USERS""" /USER=Operations-
Staff
```

TCPware SMTP uses the RFC-822 To: and CC: headers to provide the contents of the VMSmail To: and CC: fields. To enable this processing, define the logical name TCPWARE_VMSMAIL_USE_RFC822_TO_HEADER.

Note! VMSmail limits the length of its To: and CC: fields to 255 characters.

IMAP Server

The Internet Message Access Protocol (IMAP) server lets an IMAP-compliant client mail program access remote message storage as if the storage were local. TCPware’s implementation is based on IMAP Version 4, Revision 1.

IMAP and the Post Office Protocol (POP3), described in the next section, operate differently. IMAP retains the message on the server, whereas POP3 retrieves the message and stores it "off-line" on the client, thereby deleting it from the mail server. IMAP does not delete the mail message and lets you access your mail from more than one client host at a time.

IMAP was designed to:

- Be fully compatible with Internet messaging standards, such as MIME.
- Allow message access and management from more than one computer.
- Allow access without relying on less efficient file access protocols.
- Provide support for "online," "offline," and "disconnected" access modes

- Support concurrent access to shared mailboxes.
- Eliminate the need for the client software to know about the server's file storage format.

The IMAP protocol includes operations for:

- Creating, deleting, and renaming mailboxes
- Checking for new messages
- Permanently removing messages
- Setting and clearing flags
- Server-based RFC-822 and MIME parsing and searching
- Selective fetching of message attributes, texts, and portions thereof, for efficiency

For other IMAP features and how they contrast with those of POP3, see either of these web sites:

http://www.imap.org/imap.vs.pop.brief.html

http://www.imap.org/imap.vs.pop.html.

CNFNET asks if you want to use the IMAP server, and if so, provides additional prompts:

- Enter the user (account) the IMAPD (daemon) process should execute as. The default is SYSTEM. Whatever user you choose must have SYSNAM, TMPMBX, NETMBX, and SYSPRV or BYPASS privileges.
- Do you want to enable full message caching?

  The IMAP Server usually caches only the text of the last accessed message, in addition to the attributes of all messages in the currently selected folder. Enabling message caching causes the server to cache the text of all messages once seen and until the folder is closed. This can increase server performance, but requires considerably more memory.

- What is the desired logging level? The options are:

  – NONE--Does no error logging (the default)
  – ERROR--Logs errors only
  – INFO--Logs errors and informational messages
  – DEBUG--Complete debug logging

    TCPware creates a log file, TCPWARE_SPECIFIC:[TCPWARE]IMAPSERVER.LOG, when the server is started.

IMAP Mail Folders

In contrast to POP3, IMAP allows you to access server mail folders (message stores) other than INBOX. In MAIL, for example, if you create a NOTES folder, you can access mail in that folder. This NOTES folder can be in a mail file other than the default MAIL.MAI file. In fact, you can set a configuration parameter that determines the way mail folders are presented to the client so that you can use folders in these other mail files.

Your default mail directory includes a .IMAPRC file in which you can set certain configuration directives (described more fully in the *IMAP Directives File* section that follows). Among these directives is **allow-subfolders**. This directive specifies that folder names are comprised of a directory (optional), mail file, and folder. For example, the NOTES folder in MAIL.MAI is represented as mail/notes (as opposed to just notes if the directive were not set). This would distinguish it from another NOTES folder in the OLD.MAI mail file, for example, which would be named old/notes.

Each level beyond the second in this hierarchy represents a subdirectory of the default mail directory. For example, the NOTES folder in [.ARCHIVED]MAIL.MAI has the IMAP equivalent of archived/mail/notes.

Because of this folder syntax ambiguity, directory names, file names, and folders can overlap, such as the examples in the following table.

Table 16-2 IMAP Mail Folders Overlapping Syntax Examples

This mail file...	Containing this folder...	Has this IMAP equivalent...
MAIL.MAI	NOTES	mail/notes
[.MAIL]NOTES.MAI	STUFF	mail/notes/stuff
[.MAIL.NOTES]STUFFMAI	BOBS	mail/notes/stuff/bobs

Entries in the syntax can at different times be mail files, directories, subdirectories, or folders. Because of this overlap, the server must keep an internal representation of the hierarchy and mark what each level of the folder name means. This information is critical when renaming or deleting folders. Mail file definitions do not have to match.

One restriction is that a first level folder (MAIL, for example) cannot be a message store, since it represents only a file and not a mail folder. INBOX, however, is a special case. INBOX is always INBOX, cannot be deleted or renamed (a rename moves messages to the renamed folder but does not delete INBOX), and never goes away. In IMAP, INBOX is mail/NEWMAIL by default, and is hidden to the user.

(Note that you can change the mail "in-box" from INBOX to another folder by defining the **file-inbox-messages-to-folder** directive in the .IMAPRC file. See the next section for details.)

You can also access mail files in your login directory (SYS$LOGIN) by prefixing the folder name with a tilde (~). The ~ folder is reserved and cannot be used by other folders.

IMAP Directives File

Users can set certain preferences by creating a file in their default mail directory called .IMAPRC and including directives. The following table lists these directives along with their meanings. Each directive must be on its own line and in lowercase.

Table 16-3   IMAP Configuration Directives in .IMAPRC

This directive...	Does the following...
allow-subfolders	Changes the way folders are represented to the client. (See the previous section for details.)
disable-purge-reclaim	Instructs the server to not purge deleted messages upon closing a folder. Purge and reclaim are enabled by default.
file-inbox-messages-to-folder *folder*	Moves seen messages from INBOX to the specified folder upon closing INBOX.
check-new-mail-timer *delta_time*	Specifies, using OpenVMS delta time format, how often the server checks a folder for new mail. (Note that checking for new mail is time consuming for large folders.) The default is **0::30** (30 seconds).
check-new-folder-timer *delta_time*	Specifies, using OpenVMS delta time format, how often the server scans the user's directories for folders. The default is **0 :2** (2 minutes).
case- insensitive-folders	Specifies that folder names are case-insensitive. Otherwise, two folders with the same name but with different cases could become inaccessible to the IMAP client. Newly created folders are created in uppercase.

IMAP State Information Files

The IMAP server includes files created in the user's mail directory where it maintains state information, as shown in the following table.

Table 16-4   IMAP State Information Files

This file...	Stores...
.MAILBOXLIST	Folders to which the user subscribes.
.NEWMAILBOXES	List of folders known to be empty. OpenVMS MAIL deletes folders once it deletes the last message, so that the server must "remember" these folders.
*.MAI-UID	UIDVALIDITY information for folders in a mail file. Persistant UIDs allow clients to operate in “offline” and “disconnected” modes, and can improve performance if clients implement sensible caching schemes.

Table 16-4 IMAP State Information Files

This file...	Stores...
folder.uidvalidity	For each folder, the UIDs for all the messages. The file name is composed of the folder name and its UIDVALIDITY code.

POP3 Server

The Post Office Protocol Version 3 (POP3) is a multithreaded server that can handle up to 31 simultaneous client connections. It does not perform any mail delivery functions but simply allows clients (mostly PCs) to retrieve new mail from VMS MAIL inboxes.

CNFNET asks if you want to use the POP3 server, CNFNET FULL asks the following additional questions:

- What is the maximum number of new mail messages to return per connection?
- What is the desired logging level? The options are:
  - ERROR-- Logs errors only (the default)
  - INFO--Logs errors and informational messages
  - THREAD--Logs errors, informational messages, and detailed thread logging
  - DEBUG--Complete debug logging

    TCPware creates a log file, TCPWARE_SPECIFIC:[TCPWARE]POP3SERVER.LOG, when the server is started.
- Do you want to do a MAIL PURGE/RECLAIM operation for each mailbox after its use?

  If TRUE, this not only purges (deletes all messages in) the wastebasket folder, but releases deleted message space back to RMS for reuse. The default is TRUE.

POP3 runs on the local mail server so that client software on a PC can check the server for incoming mail and display it on the PC. The POP3 server down-loads the incoming mail on its machine to the PC, and deletes the original message. This is known as an “off-line” mail operation, where all message processing is local to the client and distinct from the server.

POP3 processes only new, incoming mail and does not manipulate folders other than INBOX.

The POP3 server first listens on TCP port 110 for a client request (in the form of a greeting) to download mail. The POP3 session then passes through the following three states:

AUTHORIZATION--The client identifies itself to the POP3 server and the server acquires resources associated with the client’s mail drop.

TRANSACTION--The client acquires resources and signs off.

UPDATE--The POP3 server releases resources and signs off.

The client sends commands to the POP3 server in each state and the server responds with an affirmative +OK (or negative -ERR) status. TCPware’s POP3 server supports all the minimal commands described in RFC 1939 (Post Office Protocol--Version 3) as well as the optional TOP command:

- USER
- PASS
- QUIT
- STAT
- LIST
- RETR (or TOP)
- DELE
- NOOP
- RSET
- QUIT
- UIDL

The APOP optional command is not supported.

For more information on the POP3 commands, see RFC 1939.

AUTHORIZATION State

After receiving a response of +OK POP3 server ready to the client's original greeting, the client sends the USER and PASS command combination to authenticate itself to the server. When authenticated, the server requests an exclusive lock on the client mail drop (or responds with -ERR unable to lock maildrop). Once the server opens the mail drop, it assigns each message a consecutive number and notes the message size in bytes.

TRANSACTION State

The client can now issue a series of STAT, LIST, RETR, TOP, DELE, NOOP, or RSET commands to acquire and process the messages:

- STAT requests a drop listing, the number of messages in and size of the maildrop
- LIST requests a scan listing for all messages or a particular message. This returns the assigned number for and size of each message.
- RETR requests the actual message text for a particular mail message. The client can optionally issue the TOP command, which requests the message header and a specified number of lines of the message body to send.
- DELE requests that a particular message be marked for deletion. (The message is not actually deleted until the session enters the UPDATE state.)
- NOOP is a dummy request for a simple +OK response; it has no other purpose.
- RSET requests to unmark all messages marked for deletion (DELE).

UPDATE State

Once all mail is processed, the client issues the QUIT command to enter the UPDATE state. Here the server releases resources, deletes the messages locally, and closes the connection. (This does not occur if for any reason a session does not terminate with a QUIT command.)

If the MAIL PURGE/RECLAIM option is enabled during POP3 configuration, the server also purges all messages in it wastebasket folder and releases deleted message space back to RMS. This is the default behavior, which does not occur if you disable this option during configuration.

Configuring SMTP Service for ALL-IN-1 Users

The TCPware mailer supports users of Compaq's ALL-IN-1 office automation environment (often referred to as ALL-IN-1 IOS or ALL-IN-1 Classic) and ALL-IN-1 MAIL via an interface to Message Router, the backbone of Compaq's MAILbus product line. This interface allows both ALL-IN-1 IOS and ALL-IN-1 MAIL users to send and receive SMTP mail. Message Router V3.1 or later is required for this feature to function properly. For information on sending and receiving SMTP mail from within ALL-IN-1 IOS and ALL-IN-1 MAIL, see the electronic mail chapter in the *User's Guide*.

Before Configuration

You must have Message Router V3.1 or later installed on your system before configuring the TCPware SMTP to Message Router (SMTP/MR) interface. If you wish to support automated conversion of WPS and DX documents in ALL-IN-1 messages to ASCII, you must also have the Message Router VMS MAIL Gateway (MRGATE) installation kit.

You do not need to install MRGATE on your system; however, certain object libraries in the MRGATE kit are needed to provide the necessary document conversion functions. The SMTP/MR gateway software functions even if the document conversion is not built. It does, however, cause all WPS and DX message bodyparts to be discarded as the ALL-IN-1 message passes through SMTP/MR.

Configuring SMTP/MR

The MR_CONFIGURE.COM command procedure in the TCPware: directory is used to configure the SMTP/MR gateway software. Execute this procedure with the DCL command:

```
$ @TCPWARE:MR_CONFIGURE
```

The command procedure presents a series of prompts. Enter a question mark (?) at any time to display more information about that prompt. The configuration command procedure prompts you for the following information:

1	Whether to display a detailed explanation before each question. It is recommended that you answer **YES** to this question the first time you run the configuration procedure.
2	The domain name of the gateway system. This is a domain-style host name used to refer to the gateway from the Internet. Be sure the name you choose is within a domain or subdomain over which you have administrative authority, and is not currently being used for another host. If your local domain is FLOWERS.COM, a reasonable choice for this domain name would be MR.FLOWERS.COM. This name is largely for internal use, and should not be needed to address mail to ALL-IN-1 users.

3	The domain name to be used for local ALL-IN-1 IOS users. This is a domain-style host name used to indicate to the TCPware mailer that it should pass the message to the Message Router for delivery to an ALL-IN-1 user. Be sure the name you choose is within a domain or subdomain over which you have administrative authority, and is not currently being used for another host. If your local domain is FLOWERS.COM, a reasonable choice for this domain name would be A1.FLOWERS.COM. Note that if you are not running ALL-IN-1 IOS, you should specify NONE.
4	The domain name to use for local ALL-IN-1 MAIL users. This is a domain-style host name used to indicate to the TCPware mailer that it should pass the message to Message Router for delivery to an ALL-IN-1 user. Be sure the name you choose is within a domain or subdomain over which you have administrative authority, and is not currently being used for another host. If your local domain is FLOWERS.COM, a reasonable choice for this domain name would be AM.FLOWERS.COM. Note that if you are not running ALL-IN-1 MAIL, you should specify NONE.
5	The Message Router mailbox name used for the gateway. This Message Router mailbox name is used by ALL-IN-1 users to send outbound SMTP mail. You are directed later to create this mailbox with the Message Router MRMAN utility. The default value of "SMTP" should be used. Both ALL-IN-1 IOS and ALL-IN-1 MAIL users use the address form *user@host@SMTP* to specify remote SMTP recipients. Each ALL-IN-1 mail utility places this outbound mail in the Message Router mailbox named SMTP. The TCPware mailer "picks up" mail from this mailbox and sends it via the normal SMTP delivery mechanism. The current version of TCPware allows both ALL-IN-1 IOS and ALL-IN-1 MAIL users to reply to messages imported with the V command or forwarded into ALL-IN-1 using an address of the form MRGATE:"A1::*user*" or MRGATE::"AM::*user*". Return addresses are translated from Message Router format to a more standard RFC-822 format in a fashion analogous to the DECnet to SMTP gateway conversion. The following logical names can be used to customize the translation: **TCPWARE_SMTP_MRGATE-NAME**--Change the name of the Message Router gateway mailbox. The default value is MRGATE. **TCPWARE_SMTP_A1_NAME**--Change the name of the ALL-IN-1 IOS gateway mailbox. The default value is A1. **TCPWARE_SMTP_AM_NAME**--Change the name of the ALL-IN-1 MAIL gateway mailbox. The default value is AM. **TCPWARE_SMTP_A1_DOMAIN**--Specify the RFC-822 domain associated with the ALL-IN-1 IOS gateway. Use the domain you specified when configuring your SMTP/MR gateway. **TCPWARE_SMTP_AM_DOMAIN**--Specify the RFC-822 domain associated with the ALL-IN-1 MAIL gateway. Use the domain you specified when configuring your SMTP/MR gateway.

6	The Message Router mailbox name used for delivery to ALL-IN-1 MAIL users. This Message Router mailbox is used to deliver inbound SMTP mail to ALL-IN-1 MAIL users. Normally this mailbox is named "AM", but the name is configurable. If your site uses a name other than AM, enter that name.
7	The Message Router mailbox name used by default when mail is sent to the domain name of the gateway system (for example, MR.FLOWERS.COM). Specify the default value, "MRGATE," to indicate the Message Router VMS MAIL Gateway mailbox.
8	The Message Router mailbox name for the local system. Specify the DECnet node name of the local system. SMTP/MR uses this name to generate addresses that are valid on remote systems.
9	The names of any null routes. Specify the name of the local DECnet node and any other nodes in a homogeneous VMScluster (including the cluster alias). Message Router routing information often includes local DECnet node names or a cluster alias which is superfluous in outbound SMTP mail. To determine the names of any null routes, use the MRMAN utility SHOW * command. Null routes have the general form: `nullname,      Replace=` The names you specify at this point are automatically stripped from addresses passing through SMTP/MR, greatly simplifying them when they pass into the SMTP world. Use a blank entry to terminate the list.
10	The SMTP address of the local postmaster, which should be the full domain-style e-mail address of the user who should receive error messages generated by SMTP/MR.
11	The password associated with this gateway’s Message Router mailbox (SMTP by default). Message Router protects each mailbox with a password to ensure privacy of mail messages.
12	Whether you wish to have messages with WPS and DX bodyparts automatically converted to ASCII. SMTP/MR can perform these conversions if it is linked against libraries provided as part of various Message Router products, notably MRGATE. If you intend to have messages in these special formats converted to ASCII, answer YES. If you answer NO, these bodyparts are not converted, and may be discarded.
13	You are then prompted for the names of several SMTP/MR configuration files. Accept the defaults for each of these file names.
14	You are then asked if you wish to generate the configuration files. If you are satisfied with all of your responses, type YES.

In the following example, which shows an MR_CONFIGURE.COM session, the local host’s DECnet node name is MIGUEL, while its TCP/IP host name is MIGUEL.FLOWERS.COM. The DECnet cluster alias is IRISDN; and other hosts in the homogeneous VAXcluster are named WHARFIN, BIGBTE, and YAYA. The e-mail address of the local postmaster is POSTMASTER@FLOWERS.COM.

```
$ @TCPWARE:MR_CONFIGURE

SMTP-MR Configuration Utility, Version V3.1

This utility creates an initial pair of databases for mapping SMTP
RFC 822-style addresses to Message Router X.400-style addresses and
back again. Only minimal mappings are created to support ALL-IN-1
users sending and receiving SMTP mail. If you need full MAILbus to
SMTP gatewaying capabilities, contact Innosoft International, Inc. at
(714) 624-7907 about their PMDF and PMDF-MR e-mail gateway products.

Important note: No changes are made to existing SMTP-MR database
information until all questions have been answered. This utility can be
aborted at any prompt by entering a CTRL/Z. The files output by this
utility may optionally be redirected to a different location so they will
have no impact on the existing SMTP-MR databases.

Do you wish to continue [Y]? Return
Do you wish to have a detailed explanation printed before each question
[N]? Return
Domain name of the gateway: MR.FLOWERS.COM
Domain name for local ALL-IN-1 IOS users [A1.FLOWERS.COM]: Return
Domain name for local ALL-IN-1 MAIL users [AM.FLOWERS.COM]: Return
Message Router mailbox name for the gateway [SMTP]: Return
ALL-IN-1 IOS mailbox known to Message Router [A1]: Return
ALL-IN-1 MAIL mailbox known to Message Router [AM]: Return
Message Router mailbox used by default [MRGATE]: Return
Local system's Message Router mailbox [MIGUEL]: Return

Local node name or other null routes [RETURN if no more]: MIGUEL
Local node name or other null routes [RETURN if no more]: WHRFIN
Local node name or other null routes [RETURN if no more]: BIGBTE
Local node name or other null routes [RETURN if no more]: YAYA
Local node name or other null routes [RETURN if no more]: IRISDN
Local node name or other null routes [RETURN if no more]:

RFC822 address of local PostMaster: postmaster@flowers.com
Password for the Message Router mailbox: flowers
Convert WPS-PLUS and DX messages to ASCII automatically [Y]? Return
SMTP to MR mapping text file [TCPWARE:TO_MR.]: Return
MR to SMTP mapping text file [TCPWARE:FROM_MR.]: Return
Gateway options file [TCPWARE:MR_OPTIONS.]: Return
SMTP-MR checklist file name [TCPWARE:SMTP-MR.CHECKLIST]: Return

All configuration questions have been answered.
Do you wish to generate the configuration files [Y]? Return
Generating SMTP to MR mapping text file...
SMTP to MR mapping text file is complete.

Generating MR to SMTP mapping text file...
```

```
MR to SMTP mapping text file is complete.

Generating the setup checklist...
Checklist file is complete.

Generating options file...
Options file is complete
***********************************************************************
*   To complete your SMTP-MR configuration, carry out the steps
*   detailed in the setup checklist  TCPWARE:SMTP-MR.CHECKLIST.
***********************************************************************
$
```

Configuring SMTP/MR Document Conversion

The DCF document conversion utility is used by SMTP/MR to convert WPS and DX message bodyparts to ASCII text. This utility is built from document conversion library routines supplied as part of the Message Router VMS MAIL Gateway (MRGATE) distribution kit. SMTP/MR can function without this utility, but cannot convert WPS and DX bodyparts to ASCII without it. Bodyparts in outbound SMTP mail that cannot be converted to ASCII are discarded.

The DCF utility is not supplied in executable form with SMTP/MR; it must be built after SMTP/MR is configured. The command procedure TCPWARE:DCF_BUILD.COM is provided to accomplish this. This procedure prompts for two items:

- The location of the save set from which to extract the necessary conversion libraries
- The name of a directory into which the libraries should temporarily be placed while the utility is being linked. Under MRGATE V3.1, the name of the save set containing the conversion libraries is MRGATE031.A.

You need not copy the saveset from the distribution media for DCF_BUILD.COM to work. For example, if you wish to access the libraries on a TK50 containing MRGATE while on a VAXstation 3100, you would specify the saveset name as MKA500:MRGATE031.A.

The following example shows how to build the DCF utility from a saveset located in the SYS$MANAGER directory:

```
$ @TCPWARE:DCF_BUILD
Directory to put libraries in [SYS$SCRATCH:]: Return
File specification of save set from which to extract libraries:
SYS$MANAGER:MRGATE031.A
Extracting libraries...
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]KOALA.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_BASE.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_TRANSLATE.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_MAIL_CONVERSIONS.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_DSAF.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]WPADOC.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]WPABASE.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]XPORT.OLB;1
Linking DCF utility...
```

```
Cleaning up...
Done
$
```

The DCF utility is never invoked interactively. It is always invoked automatically by the SMTP/MR gateway whenever it has mail containing WPS or DX bodyparts it needs to send via SMTP.

Note! You must use the A save set from MRGATE V3.1 or V3.2 to build DCF. Versions V3.3 and later do not contain the object files needed to link to DCF. If you upgrade to MRGATE V3.3, save your V3.1 or V3.2 distribution media.

Completing SMTP/MR Configuration

In addition to the SMTP/MR configuration data files, the file TCPWARE:SMTP-MR.CHECKLIST is created by the MR_CONFIGURE.COM command procedure. This file contains the steps needed to complete the SMTP/MR configuration, which include:

1	Adding the SMTP/MR gateway mailbox to your Message Router configuration. Run the MRMAN utility exactly as documented in the checklist file. The Message Router mailbox name and password must be exactly the same as you specified to MR_CONFIGURE.COM.
2	Building the WPS and DX document conversion utility. See the previous section for details on building this utility.

3	Configuring your Domain Name System (DNS) name server for SMTP/MR operation. You must add a Mail eXchanger (MX) record to your name server configuration for the following:
	• The domain name of the gateway itself (MR.FLOWERS.COM in the example in the *Configuring SMTP/MR* section)
	• The domain name used to direct mail to your ALL-IN-1 IOS users (A1.FLOWERS.COM in the example in the *Configuring SMTP/MR* section)
	• The domain name used to direct mail to your ALL-IN-1 MAIL users (AM.FLOWERS.COM in the example in the *Configuring SMTP/MR* section).
	If the host running SMTP/MR is named MIGUEL.FLOWERS.COM, the DNS Resource Records (RRs) you would use in the DNS configuration file for the domain FLOWERS.COM are:
	`MR.FLOWERS.COM.  IN  MX  0  MIGUEL.FLOWERS.COM.`
	`A1.FLOWERS.COM.  IN  MX  0  MIGUEL.FLOWERS.COM.`
	`AM.FLOWERS.COM.  IN  MX  0  MIGUEL.FLOWERS.COM.`
	For more detailed information on configuring a DNS name server, see Chapter 6, "Host Tables and DNS."

4	If you are not running a DNS name server locally, you must add additional host records to the TCPWARE:HOSTS.LOCAL file for the host names of the gateway and your ALL-IN-1 users. Using the names from the above example, and assuming that the IP address for MIGUEL.FLOWERS.COM is 128.0.0.1, you would add the following lines to TCPWARE:HOSTS.LOCAL:
	`HOST : 128.0.0.1 : MR.FLOWERS.COM : : : :`
	`HOST : 128.0.0.1 : A1.FLOWERS.COM : : : :`
	`HOST : 128.0.0.1 : AM.FLOWERS.COM : : : :`
	Note that you should:
	• Place these lines after the entry for MIGUEL.FLOWERS.COM.
	• Specify each name on a line by itself. Merging entries in the hosts file prevents the gateway from functioning properly.
	• Recompile and re-install the host tables after adding the new entries.
	For detailed information on adding entries to your host tables, see chapter 6.

5	Submitting the command procedure TCPWARE:MR_TO_TCPWARE.COM to the appropriate batch queue on your system. This command procedure runs periodically to transfer mail from the SMTP/MR Message Router mailbox (normally SMTP) to the TCPware mailer. Examine this command procedure before submitting it to ensure it runs in the batch queue and under the desired user name.

Enhanced MAILbus Support

If you need additional MAILbus support for ALL-IN-1 users beyond sending and receiving SMTP mail, contact Innosoft International, Inc., about their PMDF-MR product. PMDF-MR is designed to be a fully-functional gateway between MAILbus and the extensive list of protocols supported by their PMDF e-mail gateway product. For additional information on PMDF-MR and PMDF, contact:

Innosoft International, Inc.
1050 East Garvey Ave. South, Suite 250
West Covina, CA 91790
Phone: (818) 919-3600
Fax: (818) 919-3614
iii@innosoft.com

Configuring the SMTP-DECnet Mail Gateway

TCPware can be set up as a gateway to route mail between SMTP and DECnet-only hosts, with appropriate address translations to make the DECnet-style addresses easier for Internet hosts to interpret. To do this, you set the DECNET-DOMAIN mail parameter and add an appropriate MX record to the Domain Name Service. The addresses of DECnet mail sent out via SMTP will be rewritten such that the DECnet node name(s) appear under the DECNET-DOMAIN name in the host-part of the address. The addresses of incoming SMTP mail for hosts under the DECNET-

DOMAIN are automatically converted into DECnet addresses and delivered to the DECnet-only hosts.

DECnet-to-SMTP Mail

In the DECnet-to-SMTP direction, a VMS MAIL user on a DECnet-only host sends SMTP mail by specifying an address of the form: *node***::SMTP%"***user@host***"**

– *node* is the DECnet node name of the system running TCPware.

TCPware recognizes that the mail originated in DECnet and, if the DECNET-DOMAIN parameter is set, rewrites the originating address in the form *user@node.decnet-domain*.

For example, FLOWERS.COM has set up node HQ as a DECnet-SMTP gateway. A user named JOHN on DECnet-only node WHARFIN at FLOWERS.COM addresses mail to the Info-TCPware mailing list as: `HQ::SMTP%"Info-TCPWARE@ABC.COM"`

JOHN's DECnet return address, WHARFIN::JOHN, is rewritten by the gateway as:

```
JOHN@WHARFIN.DNET.FLOWERS.COM
```

instead of:

```
"WHARFIN::JOHN"@HQ.FLOWERS.COM
```

which some Internet mailers would have trouble parsing.

SMTP-to-DECnet Mail

For the SMTP-DECnet gateway to work in the SMTP-to-DECnet direction, other hosts on your network must be told that mail for host names under the DECNET-DOMAIN should be sent to the gateway host. If you use Domain Name Service, the easiest way to do this is to set up a wildcard MX record for the DECNET-DOMAIN. In our example, the MX record looks like this:

```
*.DNET.FLOWERS.COM.    IN        MX     0    HQ.FLOWERS.COM.
```

This MX record causes other hosts on the Internet to send mail destined for any host under DNET.FLOWERS.COM to node HQ. The gateway automatically recognizes the DECNET-DOMAIN in the host-name part of the address, rewrites the address to its DECnet form, and sends it through VMS MAIL.

If you do not use DNS, you must add a fully qualified host name for each DECnet node to your host tables. In our example, a return message to user JOHN on node WHARFIN would be addressed to:

```
JOHN@WHARFIN.DNET.FLOWERS.COM.
```

When HQ receives that message, it translates the address to its DECnet form:

```
WHARFIN::JOHN
```

and sends the message to that address using VMS MAIL.

Chapter 17

Managing TELNET-OpenVMS Server

Introduction

This chapter describes the TELNET-OpenVMS Server from the system manager's point of view. Topics include:

- TELNET logicals
- TELNET control functions
- Setting up Virtual Terminals
- TELNET options

RFCs 854 through 858, RFC 885, and RFC 1091 describe the TELNET protocol. The *User's Guide*, Chapter 11, *TELNET: Connecting to Remote Terminals*, the *Command Reference* section describes the commands you can use with TELNET.

TELNET Logicals

The logicals in Table 17-1 appear in the TCPWARE:TELNET_CONTROL.COM file.

CAUTION! Editing the TELNET_CONTROL.COM file is not recommended, since each TCPware installation replaces this file. If you want to define a special purpose logical, do so in another place, such as in your system startup file.

Table 17-1   TELNET Logicals

Logical	Description
TCPWARE_TELNETD_INTRO_MSG	Define this logical as a special message that appears whenever a user reaches the host through TELNET. If the system logical name table contains a value for this logical, the Server sends the equivalence string to the peer before the standard login sequence starts. Use this logical to issue warnings such as `Authorized Use Only` for remote logins.
TCPWARE_TELNETD_DEFCHAR	Define this logical to set up the default terminal characteristics for TELNET sessions. You can thereby avoid having to change the SYSGEN TTY_DEFCHAR and TTY_DEFCHAR2 fields system-wide. (See *Virtual Terminals*.)
TCPWARE_TELNET_WINDOW	If defined, specifies the window size that the TELNET Server offers to the peer. By default, this value is 4096. If the TCPWARE_TELNET_WINDOW logical's value is less than 512, TELNET uses 4096.
TCPWARE_TELNETD_FLAGS	This logical has several purposes. The default value is 1. Table 31-2 describes the bit options. Setting either bit 0 or 1 can improve Server performance and reduce system processing overhead. Note, however, that doing so means that you are not adhering to the TELNET protocol.

Table 17-2   TCPWARE_TELNETD_FLAGS Logical Bit Description

Bit...	(Mask)	Set to...	Description
0	1	0	Inserts a <NULL> character following a <CR> character that is not followed by an <LF> character. (Ignored if bit 1 is set to 1.)
		1 (default)	Inserts nothing after a <CR> character that is not followed by an <LF> character.
1	2	0 (default)	Server does not run in raw mode. It doubles <IAC> data characters as required by the TELNET protocol.

Table 17-2 TCPWARE_TELNETD_FLAGS Logical Bit Description (Continued)

Bit...	**(Mask)**	**Set to...**	**Description**
		1	Ignores bit 0: Sends all characters without special processing, doubling of characters, or inserting <NULL> after <CR>.
7	128	`0` (default)	Incoming TELNET sessions are REMOTE.
		1	Incoming TELNET sessions are LOCAL (compatible with earlier TCPware versions).
8	256	0 (default)	Tries to return the peer's hostname in the NTA terminal's TT_ACCPORNAM field.[*] Otherwise uses the IP address.
		1	Ignores bit 9: Adds the port number to the IP address, such as `192.168.95,1094` (compatible with an earlier TCPware).
9	512	0 (default)	Tries to return the peer's hostname in the NTA terminal's TT_ACCPORNAM field.[*] Otherwise uses the IP address.
		1	Adds the port number to either the hostname or IP address, such as `bart.nene.com,1094` or `192.168.95,1094`. (Ignored if bit 8 is set to 1.)
The combination of bit 8 and 9 settings has the following effect on TT_ACCPORNAM:[*]			
Bit 8	Bit 9	**Returned `Remote Port Info` (TT_ACCPORNAM Field)**	
0	0	*host* or *address* (default): `bart.nene.com` or `192.168.95`	
0	1	*host,port* or *address,port*: `bart.nene.com,1094` or `192.168.95,1094`	
1	0	*address,port*: `192.168.95,1094`	
1	1	*address,port*: `192.168.95,1094`	
[*] The TT_ACCPORNAM field is limited to 63 bytes; if the value is too long, the port number may be truncated or even dropped.			

Virtual Terminals

Virtual terminals (VTAs) allow users to disconnect from a physical terminal without terminating a process – the process remains active on a virtual terminal. Virtual terminals are used to reconnect to a process when the connection is lost, and to maintain sessions on more than one disconnected terminal.

To set up the TELNET Server to support VTAs:

<table>
<tr><td>1</td><td>Set up VTA devices as follows:

<code>VAX:</code>

<code>$ RUN SYS$SYSTEM:SYSGEN</code>
<code>SYSGEN> CONNECT VTA0: /NOADAPTER /DRIVER=TTDRIVER</code>
<code>SYSGEN> EXIT</code>

<code>ALPHA:</code>

<code>$ RUN SYS$SYSTEM:SYSMAN</code>
<code>SYSMAN> IO CONNECT VTA0 /NOADAPTER-</code>
<code>_SYSMAN> /DRIVER=SYS$LOADABLE_IMAGES:SYS$TTDRIVER.EXE</code>
<code>SYSMAN> EXIT</code></td></tr>
<tr><td>2</td><td>Edit the TELNET_CONTROL.COM file to define the TCPWARE_TELNET_DEFCHAR logical (see TELNET Logicals):

<code>$ DEFINE/SYSTEM TCPWARE_TELNETD_DEFCHAR 402657952,135174</code>

TCPWARE_TELNETD_DEFCHAR permits TELNET server devices to override the values in the SYSGEN TTY_DEFCHAR and TTY_DEFCHAR2 parameters, if they are not set up correctly for TELNET sessions. These SYSGEN parameters apply to all terminals and only take effect after a reboot.

In the above logical definition:

• <code>402657952</code> (<code>%X180012A0</code>) is the VMS default value for TTY_DEFCHAR.
• An optional value <code>135174</code> (<code>%X0021006</code>) for TTY_DEFCHAR2 is appended.
• The default value for TTY_DEFCHAR2 is normally <code>%X0001002</code>, which is TT2$M_SECURE combined with TT2$M_AUTOBAUD.
• The value <code>135174</code> (<code>%X00021006</code>) reflects the addition of TT2$M_DISCONNECT (<code>131072</code>, or <code>%X00020000</code>) and TT2$M_HANGUP (<code>4</code>).

(See the SYS$LIBRARY:TTDEF.H file and SYS$LIBRARY:TT2DEF.H file for bit definitions for TTY_DEFCHAR and TTY_DEFCHAR2, respectively.)

• TT2$M_DISCONNECT is needed to allow disconnected virtual terminals.
• TT2$M_HANGUP is also recommended, since you want the TELNET sessions to disconnect when you log out.</td></tr>
</table>

3	Restart TELNET for the changes to take effect: `$ @TCPWARE:RESTART TELNET` When a client opens a session to the server set up with VTAs and the virtual terminal disconnects, the opening sequence might be as follows:

```
Username:
Password:
     You have the following disconnected process:
Terminal    Process name   Image name
VTA2:_      VTA2:          (none)
Connect to above listed process [YES]: Return
Connecting to terminal _VTA2:
```

Options

The Client-TELNET utility and TELNET Server support the following options:

ECHO	END-OF-RECORD	REMOTE-FLOW-CONTROL
TERMINAL-SPEED	TERMINAL-TYPE	TRANSMIT-BINARY
SUPPRESS-GO-AHEAD	WINDOW-SIZE	

The Client and Server negotiate options using the TELNET protocol WILL, WONT, DO, and DONT commands.

ECHO

The ECHO option enables or disables echoing data received over the network. You can configure each side of a connection independently. The initial default is no echoing.

The Client supports enabling and disabling of echoing for characters it sends over the network. It does not support echoing for characters it receives over the network. The Server supports enabling and disabling of echoing for characters it receives over the network. It refuses any attempts to have the Client echo characters that the Client receives over the network.

The user’s or interactive process terminal sets the OpenVMS terminal device driver TT$M_NOECHO characteristics depending on the echo requirements. Both the Client and Server attempt to negotiate for the Server to echo the characters it receives over the network from the Client.

END-OF-RECORD

Use the END-OF-RECORD option during IBM 3270-class terminal emulation. This option affirms that the client and server both use the TELNET end-of-record character. Use this character to delimit TELNET screens.

REMOTE-FLOW-CONTROL

The Remote Flow Control Option (RFC 1372) is supported on both the Client and Server to enable or disable local flow control on the client, or the handling of `Ctrl/S` and `Ctrl/Q` keystrokes to stop and resume TELNET transmission, respectively. These controls are usually processed locally by the terminal driver and are not sent to the remote server.

The SET LOCAL_FLOW_CONTROL and SET NOLOCAL_FLOW_CONTROL commands are provided on the Client if the remote server does not support the Flow Control Option. By specifying SET NOLOCAL_FLOW_CONTROL, the flow control characters are passed to the remote server and are not processed locally.

The default flow control setting depends on the TT$V_TTSYNC value for the terminal. You can set "TTSync" mode (local flow control) outside of TELNET by using the DCL SET TERMINAL /TTSYNC command, or set "No TTSync" mode (server flow control) by using the DCL SET TERMINAL /NOTTSYNC command; some full-screen editors also set these modes. However, if you are inside TELNET, SET NOLOCAL_FLOW_CONTROL can force the terminal into "No TTSync" mode for a particular connection.

SUPPRESS-GO-AHEAD

The SUPPRESS-GO-AHEAD option enables or disables sending the TELNET "go-ahead" (GA) control function. You can configure each side of a connection independently. The initial default is not to suppress go-aheads.

Both the Client and Server support negotiating this option. Regardless of the option’s state, both ignore the GA control function if they receive it and never transmit it. The user can send this option by entering the SET GA or SEND commands in the Client. Both the Client and Server attempt to negotiate for GA suppression.

TERMINAL-SPEED

The Terminal Speed Option (RFC 1079) is useful for applications that may want to adjust some actions based on the baud rate at which a user connects. The Server does an equivalent of a DCL SET TERMINAL/SPEED command. This is supported on the Client and Server.

TERMINAL-TYPE

TELNET uses the TERMINAL-TYPE option to negotiate the type of terminal used. TELNET uses this option if both the client and server support the option and are willing to use it. This option tells the TELNET server what type of terminal the client has.

TRANSMIT-BINARY

The TRANSMIT-BINARY option enables or disables the full eight-bit ASCII character set. You can configure each side of a connection independently.

The initial TELNET protocol default is to use the seven-bit ASCII character set. However, the peer TELNET implementation can strip the eighth bit unless you use the TRANSMIT-BINARY option. For the Client, use the SET BINARY command to enable full eight-bit mode.

The Client and Server always transmit the full eight-bit ASCII character set over the network, regardless of the option's status.

When you request the Client to enable the TRANSMIT-BINARY option, it sets the OpenVMS terminal device driver TT$M_EIGHTBIT characteristics. When you request the Server to enable this option, it does not change the TT$M_EIGHTBIT characteristic.

Depending on the state of the EIGHTBIT terminal option, OpenVMS may strip the eighth bit. OpenVMS strips the eighth bit on input if the terminal attribute is NOEIGHTBIT. However, OpenVMS does not strip this bit on output.

WINDOW-SIZE

The Window Size Option (RFC 1073) is also known as the Negotiate About Window Size (NAWS) option. The client and server negotiate sending the window size information using the standard TELNET WILL/DO/DONT/WONT mechanism. If the client and server agree, the client may send a subnegotiation to convey the window size. If the client's window size later changes, the client may send a subsequent subnegotiation. This is supported on the Client and Server.

Control Functions

The TELNET protocol defines several control functions. Some of these functions include interrupting a process, aborting output, and erasing a character or line.

The Client sends these functions with the SEND command, or if the user types characters defined with the SET commands. The Client ignores these functions if it receives them.

The Server never sends these functions. The Server replaces the received functions with an OpenVMS character sequence as shown in Table 17-3. The exception is the "are you there" (AYT) function that returns the BELL character (ASCII 7).

The Server recognizes the control functions listed in Table 17-3; it ignores all other control function.

Table 17-3 Supported TELNET Control Functions

TELNET control function...	Recognizes OpenVMS character sequence...
Interrupt Process (IP)	Ctrl/Y
Abort Output (AO)	Ctrl/O
Erase Character (EC)	DELETE
Erase Line (EL)	Ctrl/U

Exiting Status

To exit TELNET use the EXIT command or type `Ctrl/Z`

TELNET exits with the last error status, if any. DCL command procedures can use the $STATUS and $SEVERITY symbols to test for success or failure of the TELNET commands issued. A success status indicates that all commands succeeded. A warning, error, or severe status indicates that one or more commands failed to execute, either because of syntax errors or because of operational problems.

When possible, the status code is a System Service (defined in $SSDEF), RMS (defined in $RMSDEF), or shared (defined in $SHRDEF) status. In some cases, status codes are TCPware private codes with a facility number of 1577.

PART VI Managing Security

Chapter 18

Managing TCPware Security

Introduction

This chapter presents the security features available with TCPware.

TCPware security features can be divided into two parts: independent, free-standing security features, and those inherent within products. The free-standing security features include:

Incoming Access Restrictions	Kerberos Services	Packet Filtering
Outgoing Access Restrictions	IP Security Option (IPSO)	Token Authentication

TCPware products for which security features are available include:

Berkeley R Services	NFS-OpenVMS Server	FTP-OpenVMS
DECwindows	Remote Copy Program (RCP)	TELNET-OpenVMS

Security Tips

Here are some general tips to maintain system security on your TCPware system:

1	Implement an aggressive password strategy for users and especially for privileged accounts. This means long passwords and frequent password changes.
2	Disable all unused accounts, especially privileged ones that are predefined with the operating system (such as field test accounts).
3	If a break-in is in progress, use one of the DEBUG commands in the Network Control Utility (NETCU) to capture the packets. However, be advised that monitoring might be construed as an invasion of privacy.

4	Carefully consider the system announcement or welcome messages. Do not welcome people to the site. Instead, make it clear that the system is for authorized users only, and that all users may be monitored for security reasons. For details, see a recent book on computer security, such as Cheswick, William R. and Steven M. Bellovin, *Firewalls and Internet Security – Repelling the Wily Hacker.*

5	Subscribe to CERT (Computer Emergency Response Team) announcements by e-mail. If you believe that your system was compromised, contact the CERT Coordination Center or your representative in Forum of Incident Response and Security Teams (FIRST). If you wish to send sensitive incident or vulnerability information to CERT staff by e-mail, we strongly advise that the E-mail be encrypted. The CERT Coordination Center can support a shared DES key, PGP (public key available through anonymous FTP on `info.cert.org`), or PEM (contact CERT staff for details).
	
	Here are the contacts at CERT:
	
	Internet E-mail: `cert@cert.org`
	Telephone: +1 412-268-7090 (24-hour hotline)
	CERT personnel answer 8:30 AM–5:00 PM EST (GMT-5) or
	EDT (GMT-4), and are on call for emergencies during other hours.
	Fax: +1 412-268-6989
	Postal address:CERT Coordination Center
	Software Engineering Institute
	Carnegie Mellon University
	Pittsburgh, PA 15213-3890
	USA
	
	The following documents are also available on the Web using the following URLs:
	
	`ftp://info.cert.org/pub/tech_tips/security_info`
	`ftp://info.cert.org/pub/tech_tips/anonymous_ftp_abuses`
	`ftp://info.cert.org/pub/tech_tips/packet_filtering`
	
	CERT posts advisories and bulletins on the `comp.security.announce` USENET newsgroup. If you wish to have future advisories and bulletins mailed to you or to a mail exploder at your site, send mail to `cert-advisory-request@cert.org`.
	
	Past advisories, CERT bulletins, information about FIRST representatives, and other information related to computer security are available by anonymous FTP from `info.cert.org`. (CERT is a service mark of Carnegie Mellon University.)

Independent Security Features

This section describes the security features that you can use with more than one TCPware product. These include incoming and outgoing access restrictions, packet filtering, Kerberos Services, IP Security Option, and Token Authentication.

Incoming Access Restrictions

With incoming access restrictions, the system imposes restrictions on the remote hosts having access to local services.

Manage incoming access restrictions using the following NETCU commands:

ADD ACCESS_LIST	REMOVE ACCESS_LIST	MODIFY SERVICE
ADD SERVICE	SHOW ACCESS_LISTS	

To manage incoming access restrictions, see Chapter 19, *Access Restrictions*.

Outgoing Access Restrictions

With outgoing access restrictions, you restrict access to outside services such as FTP or TELNET for local users. You can base restrictions on a user's ID (UIC or rights identifiers), or the destination address or port (service). Since the system checks only outgoing access restrictions with active open connections and not on each and every datagram, it involves relatively low system overhead.

Manage outgoing access restrictions using two commands in the TCPware Network Control Utility (NETCU):

- SET OUTGOING_ACCESS_RESTRICTIONS
- SHOW OUTGOING_ACCESS_RESTRICTIONS

To manage outgoing access restrictions, see Chapter 35, *Access Restrictions*.

Packet Filtering

Packet filtering restricts the datagrams that may be received on a network interface. The system drops all datagrams it denies entry. This software-based filtering allows you to filter datagrams by:

- Protocol (IP, TCP, UDP, or ICMP)
- Source or destination address
- Destination port (TCP or UDP) or ICMP message type

Packet filtering prevents a site from receiving datagrams from certain networks or hosts. For example, a site might wish to restrict incoming access from the rest of the Internet, but allow local users to have full access to Internet resources.

Use packet filtering only when absolutely necessary since the system must scan the packet filter for each datagram. If there is a question whether to use packet filtering or access restrictions on incoming datagrams, packet filtering is more complete, since it covers all connections. However, packet filtering requires more overhead than incoming access restrictions.

Manage packet filtering using the following NETCU commands:

- SET FILTER or SET NOFILTER
- SHOW FILTER

To manage packet filtering, see Chapter 20, *Packet Filtering*.

Kerberos Services

TCPware provides the following Kerberos Services:

Kerberos Server	Kerberos for RCP	Kerberos Administration Server
Kerberos User Commands	Kerberos for RLOGIN	Kerberos for TELNET
Kerberos Management Commands	Kerberos for RSH	

Some Terms

Some of the terms commonly associated with Kerberos include:

Term	**Description**
Principal	Kerberos refers to clients and servers as principals. Kerberos assigns each principal a name, in the general format: **name.instance@realm** -*name*, for clients, is the user's login name; for servers, is the name of the service provided, usually `rcmd`. -*instance*, for clients, is usually omitted and is not necessary; for Kerberos administrators, the value is **`admin`**; for servers, *instance* identifies the machine name of the application server that has Kerberos authentication support. For example, if the `rlogin` server on `merlin` has Kerberos authentication support, the principal would have the following format: **rcmd.merlin@your_realm** -*realm* is associated with all principals in a Kerberos database and is the name of a group of machines, such as those on a LAN; *realm* identifies the Kerberos domain. You can omit the last two components from some principals. For example, a possible principal could be `jones` (for user Jones in the local domain) or `jones@daisy.com` for user Jones in the `daisy.com` domain. A possible principal could also be `rcmd.merlin` (for the `rlogin` server in the local domain) or `rcmd.merlin@daisy.com` (for the `rlogin` server on `merlin` in the domain `daisy.com`).

Term	Description
Ticket-granting ticket	Contains an encrypted form of the user's Kerberos password. Use it to obtain application service tickets from the Kerberos Server. You cannot use Kerberos authentication without first having this ticket-granting ticket. The ticket-granting ticket has an associated lifetime that the Kerberos Server specifies. This lifetime is generally eight hours. You can use the same ticket over and over again, until you no longer need the ticket or it expires.
Application service ticket	The Kerberos protocol application uses service tickets to verify a client's identity to an application server. The Kerberos Server encrypts the service ticket with the application server's private key. Only that application server can decrypt the service ticket.
Authenticator	The Kerberos protocol uses authenticators to prevent eavesdroppers from stealing a ticket. The client sends a new authenticator with each service request. An authenticator consists of the client's name, client's IP address, and a timestamp showing the current time. The server uses the information in the authenticator to confirm that the rightful owner presents the accompanying ticket. For this to be true, the client and server must synchronize their clocks. One way is through the Network Time Protocol (NTP).

Note that a service ticket and authenticator only accompany the request for a service; you do not use them for data exchange once you initiate the service.

Kerberos Process

There are three main steps in the Kerberos process. The user:

- Gets a ticket-granting ticket from the Kerberos Server.
- Employs this ticket-granting ticket to get the application service ticket.
- Requests service by presenting the service ticket and an authenticator to the application server.

The Kerberos process uses tickets, authenticators, and messages. These elements provide specific encrypted information about clients and servers. Kerberos uses keys to encrypt and decrypt tickets, authenticators, and messages.

Some things to remember about tickets and authenticators:

- A client must have service tickets for access to any application server.
- The client cannot read service tickets since the Kerberos Server encrypts them with the private key of the application server. Every ticket has a session key.

- Kerberos requires a new authenticator from the client each time the client starts a new connection with a server.
- The encrypted service ticket and authenticator contain the client’s network address.
- The service ticket and authenticator have a short lifetime (generally five minutes).

Command Users

Regular users and Kerberos administrators use the Kerberos commands in TCPware’s Network Control Utility (NETCU). Regular users and Kerberos administrators can also use parts of the KADMIN Server.

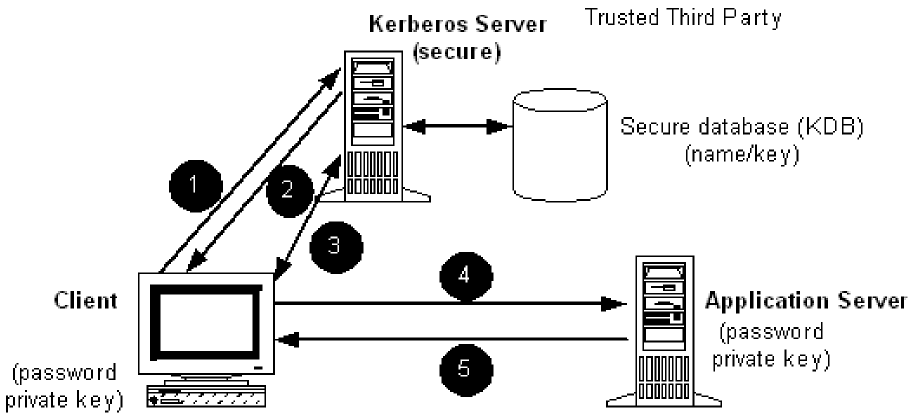
- Regular users can:
  - Get tickets.
  - Manage (show and remove) their tickets.
  - Change their Kerberos passwords.
- Kerberos administrators can:
  - Create the Kerberos database (KDB).
  - Manage the KDB.
  - Control remote access to the KDB.
  - Add Kerberos users remotely.
  - Show Kerberos users remotely.
  - Change a Kerberos user's password remotely.

Typical Session

The following list and Figure 18-1 present a typical implementation of the Kerberos process:

1	The client submits a request to the Kerberos Server to obtain a ticket-granting ticket (TGT). The Kerberos Server consults the Kerberos database (KDB) to get the user's Kerberos password, and encrypts it.
2	The Kerberos Server sends back the encrypted password in the TGT. When the client receives the TGT, it asks the the user for his Kerberos password, encrypts the password, and compares it with the password in the TGT. This is how the user authenticates himself to the Kerberos Server.
3	The client uses the TGT to apply for service tickets so that it can use specific applications. Each service ticket proves the client's identity to an application server.
4	The client presents the service ticket to the application server for authentication. The application server decrypts part of this ticket to check its authenticity.
5	If the application server deems the service ticket to be authentic, it applies the access control it previously defined for that client. If the server cannot decrypt the service ticket, or the service ticket expired or is not authentic, it does not authenticate the client.

Figure 18-1 Typical Kerberos Session Sequence



Kerberos Server

Kerberos is an authentication system for networks designed as part of Project Athena at the Massachusetts Institute of Technology. Kerberos provides network security by regulating user access to networking services. Kerberos uses a set of keys and encrypted tickets to authenticate users.

In a Kerberos environment, at least one system runs the Kerberos Server. Keep this system secure. This trusted server provides authentication services to prove that the requesting user is genuine. Another name for the Kerberos Server is the Key Distribution Center (KDC).

The system manager assumes that other servers on the network and all clients are untrustworthy. For the Kerberos protocol to work, all systems relying on Kerberos must trust only the Kerberos system itself.

The Kerberos Server maintains a secure database, the Kerberos database (KDB), listing the names and private keys of all clients and servers allowed to use the Kerberos Server. Kerberos assumes that all users keep their passwords secure.

User Commands

TCPware provides commands for users to get a ticket-granting ticket, show the ticket-granting ticket and any service tickets, and remove all Kerberos user tickets. Kerberos tickets allow use of Kerberos applications.

TCPware implements the following user commands through NETCU:

GET TGT	SET KERBEROS_PASSWORD	REMOVE TICKETS	SHOW TICKETS

For details on the Kerberos user commands, see Chapter 4, *Kerberos User Commands*, in the *User's Guide*.

Management Commands

TCPware provides the system manager with an interface to the Kerberos database (KDB) using the following commands:

ADD KACL	MODIFY KDB	MODIFY KERBEROS USER
ADD KDB	REMOVE KACL	SET MASTER_PASSWORD
CREATE KDB	REMOVE KDB	SHOW KERBEROS USER
CREATE SRVTAB	SHOW KACL	STASH MASTER_PASSWORD
DUMP KDB	SHOW KDB	ADD KERBEROS USER
LOAD KDB		

Kerberos Server management also includes setting certain configuration logicals.

For details on the Kerberos management functions, see Chapter 38, *Managing Kerberos*.

Administration Server

The Kerberos Administration Server allows remote administration of the Kerberos primary server database. It also allows Kerberos users to change their Kerberos passwords.

Authentication for RCP

You can configure the RSH server, which also handles Remote Copy Program (RCP) requests, to require Kerberos authentication requests, to allow both Kerberos authentication and standard authentication requests, or to disallow any Kerberos authentication requests. RCP users must authenticate themselves to the Kerberos Server before using Kerberos authentication for RCP.

Authentication for RLOGIN

You can configure the RLOGIN server to require Kerberos authentication requests, to allow both Kerberos authentication and standard authentication requests, or to disallow any Kerberos authentication requests. RLOGIN users must authenticate themselves to the Kerberos Server before using Kerberos authentication for RLOGIN.

Authentication for RSH

You can configure the remote shell (RSH) server to require Kerberos authentication requests, to allow both Kerberos authentication and standard authentication requests, or to disallow any Kerberos authentication requests. RSH users must authenticate themselves to the Kerberos Server before using Kerberos authentication for RSH.

Authentication for TELNET

You can configure the TELNET server to require Kerberos authentication requests, to allow both Kerberos authentication and standard authentication requests, or to disallow any Kerberos authentication requests. TELNET users must authenticate themselves to the Kerberos Server before using Kerberos authentication for TELNET.

IP Security Option

The IP Security Option (IPSO) is a standard for preventing a system from receiving or transmitting unauthorized datagrams. It was developed for the U.S. Department of Defense and conforms to RFC 1108, *U.S. Department of Defense Security Options for the Internet Protocol*. IPSO incorporates both a Basic Security Option and an Extended Security Option. TCPware supports both of these options.

To manage IPSO, see Chapter 23, *IP Security Option (IPSO)*.

Token Authentication

Token Authentication allows you to set additional security restrictions on your FTP, TELNET, RLOGIN, and SET HOST logins. You can set up Token Authentication through TCPware's Access Control Encryption Client (ACE/Client) on the OpenVMS host, which communicates with Security Dynamics' ACE/Server on a UNIX or Windows NT host. The authentication takes place through a physical SecurID[token "smart card" that you use to provide the ACE/Server with the necessary login information.

To manage Token Authentication, see Chapter 21, *Managing Token Authentication*.

Component Security

This section describes the TCPware security features available with particular TCPware products.

Berkeley R Commands

The Berkeley R Commands implement security at TCPware configuration and later through Service Access Lists and host equivalence files. The R Services use standard TCPware and OpenVMS security facilities to ensure that only authorized hosts and users have access to the TCPware host.

TCPware implements Berkeley R Command security through:

Configuration	During TCPware configuration, you first select whether you want to enable the `login`, `shell`, and `exec` services. For `login` service, you also specify whether you want NORMAL or SECURE login authorization (the default is SECURE). The `login` service is used for RLOGIN, while `shell` or `exec` is used for RCP, RMT, and RSH (`exec` if you specify a username and password).
Service Access Lists	If desired, create a Service Access List to control which hosts, group of hosts, or network can have access to the service.

Host equivalence files	Host equivalence files allow a set of remote hosts access to the server. They contain single-line entries for each authorization. The system manager can set up a TCPWARE:HOSTS.EQUIV file on the system level. Each individual client user can also set up a SYS$LOGIN:.RHOSTS file. This allows equivalent access beyond that specified in the TCPWARE:HOSTS.EQUIV file. The format for SYS$LOGIN:.RHOSTS and TCPWARE:HOSTS.EQUIV file entries is identical.

See Chapter 15, *Managing R Commands*, for details on host equivalence files.

The typical Berkeley R Commands server security steps are that the server:

1	Checks that the Service Access List (if any) is configured to protect the desired service.
2	Checks that the client's port number is in a reserved range.
3	Checks the password file for an entry for the supplied username.
4	Searches the `/etc/hosts.equiv` (on UNIX systems) or TCPWARE:HOSTS.EQUIV (on OpenVMS systems) file for the hostname and username.
5	Searches the `.rhosts` (on UNIX systems) or .RHOSTS (on OpenVMS systems) file in the user's home directory for the hostname and username. Note that if Kerberos authentication is used, the server searches the `.klogin` (on UNIX systems) or .KLOGIN (on OpenVMS systems) file in the user's home directory for the user's Kerberos principal name.
6	Prompts for a password if SECURE `login` is specified (for RLOGIN) and there is a match-up in the .RHOSTS or HOSTS.EQUIV file, or a username and password if NORMAL `login` is specified (for RLOGIN) without a match-up in the .RHOSTS or HOSTS.EQUIV file. If the user is prompted for the password, and the TCPware ACE/Client is enabled and the user designated for Token Authentication, the user is also prompted for the PASSCODE.
7	Grants or rejects access depending on the server configuration and authorization results: • Grants access for `shell`, `exec`, or NORMAL `login` service without a login prompt if there is a match-up in the .RHOSTS or HOSTS.EQUIV. file, or for SECURE `login` service if the password entered is authorized. • Rejects access if the server is configured for `shell`, `exec`, or SECURE `login` service and there is no matchup in the .RHOSTS or HOSTS.EQUIV files, or for NORMAL `login` service if the password entered fails authorization.

Additional password protection is available using Kerberos authentication. This feature is available for the RCP, RLOGIN, and RSH Berkeley R Commands.

For details on Service Access Lists, see the ADD ACCESS_LIST in the *NETCU Command Reference*.

DECwindows

TCPware provides the following security for the DECwindows Transport Interface:

Target display host configuration	Configure the target display host to allow incoming X Window System applications from the OpenVMS system host.
Displaying on the local host	Bring up the `Security` option under the Session Manager's `Options` menu.
Displaying on the remote host	Configure "security" on the remote host to allow incoming connections on the currently active session.

To manage DECwindows security, see Chapter 44, *DECwindows Transport Interface*.

FTP-OpenVMS

FTP-OpenVMS Server provides the following security functions and options:

Table 18-1 Security Functions and Options

Functions and Options	**Description**
Passwords	Similar to DECnet, you cannot use FTP to log in to multiple-passworded accounts. If the TCPware ACE/Client is enabled and the user is designated for Token Authentication, the user must provide the PASSCODE at the password prompt.
Directory access restrictions	Server-FTP lets you define the TCPWARE_FTP_ROOT logical for directory access restrictions on a system-wide basis, the TCPWARE_FTP_ANONYMOUS_ROOT logical for ANONYMOUS user directory access restrictions, and the TCPWARE_FTP_*username*_ROOT logical for directory access restrictions for specific user accounts.
Log file	The FTPSERVER_DTP.LOG file contains information about files transferred during the FTP session. Examining this file helps to isolate security problems.
Idle timeout	If the control connection (other than during a data transfer) is idle for more than 10 minutes, the system aborts the connection.
SYLOGIN.COM	This procedure and user account login command procedures can contain commands that cause the login to fail.

Table 18-1   Security Functions and Options (Continued)

Functions and Options	Description
FTP ANONYMOUS Accounts	FTP-OpenVMS supports ANONYMOUS accounts whereby the user can log in using the ANONYMOUS login and the GUEST password. You set up ANONYMOUS accounts using the AUTHORIZE utility. ANONYMOUS users only have read-only access unless you define the TCPWARE_FTP_ANONYMOUS_RIGHTS logical to allow write, rename, and delete access rights. In addition, the TCPWARE_FTP_ANONYMOUS_ROOT logical allows you to restrict ANONYMOUS users to files in the root directory and its subdirectories.
TCPWARE_FTP_LOGFILE	Define this system logical name to specify the log file name if you suspect break-in attempts through the server.
TCPWARE_FTP_421_REPLY	This logical defines a message to send when a user you want to prevent from logging in connects to the server. After sending the message, the connection closes.

To manage FTP-OpenVMS Server security, see Chapter 20, *Managing FTP-OpenVMS*.

NFS-OpenVMS Server

The NFS-OpenVMS Server provides several features that maintain the integrity of the OpenVMS file system:

PROXY database	The Server requires that the local system must register any user attempting access to OpenVMS files. You do this through the PROXY database when you configure the Server, and through later modifications as needed.
EXPORT database	You must export an OpenVMS directory for an NFS user to have access to it. The Server does this through the EXPORT database when you configure the Server, and through later modifications as needed.

You can take the following additional system security measures:

- Assign an NFS rights identifier to further restrict file access
- Require all RPC requests to originate from privileged ports
- Restrict all remote mounts to the NFS superuser only
- Restrict mounts only to explicit directories and not their subdirectories
- Require the PROXY database to define the mount requester’s identification

See Chapter 13, *Managing NFS-OpenVMS Server*.

Remote Copy Program

Passwords can be a problem in RCP since the /PASSWORD qualifier requires entry of plain text that someone on the network can intercept.

You can prevent users from having to specify the /USER, /PASSWORD, or /TRUNCATE qualifier with the RCP command. Check that remote hosts include your hostname entry in their host equivalence files (such as the `/etc/hosts.equiv` file in UNIX systems). On OpenVMS hosts, the TCPWARE:HOSTS.EQUIV or SYS$LOGIN:.RHOSTS file serves as the host equivalence file to permit remote hosts to log in.

Kerberos password protection is also available for the RCP service.

To manage RCP security, see Chapter 15, *Managing R Commands*. To manage Kerberos authentication for RCP, see Chapter 22, *Managing Kerberos*.

TELNET-OpenVMS

The TELNET-OpenVMS Server provides the TCPWARE_TELNETD_INTRO_MSG option. With this logical, you can define a special message that appears whenever a user attempts access to the host through TELNET. You can use this logical to issue warnings such as "Authorized Use Only" for remote logins.

If the TCPware ACE/Client is enabled and the user is designated for Token Authentication, the user is also prompted for the PASSCODE in addition to the username and password.

Kerberos password protection is also available for the TELNET service.

To manage TELNET-OpenVMS security, see Chapter 17, *Managing TELNET-OpenVMS Server*. To manage Kerberos authentication for TELNET, see Chapter 22, *Managing Kerberos*.

Chapter 19

Access Restrictions

Introduction

This chapter describes how to manage incoming access restrictions:

- Incoming access restrictions are used to restrict the hosts (or groups of hosts) allowed access to TCP-based services that the TCPware master server initiates. You can use incoming access restrictions to help secure your host by restricting which hosts have access to its services.
- Outgoing access restrictions apply to outgoing TCP connections made from the TCPware host. You can use them to limit the TCP-based services that TCPware host users can use on other systems. You can base these restrictions on user rights identifiers, or destination IP addresses or ports.

You can also use outgoing access restrictions to log or alarm outgoing connections. Log means that a record is written to the TCPWARE:NETCP.LOG file, which identifies the user and the destination of the connection. Alarm means that this same message is sent to OPCOM. Even if you do not restrict the connections users can make, you might want to log or alarm all attempts.

If the process attempting to open a connection has SYSPRV or BYPASS privilege or is running under a system UIC, outgoing access restrictions are not applied. However, logging and alarming entries are still honored.

Incoming Access Restrictions

Permit or deny incoming access using access lists controlled by commands in the Network Control Utility (NETCU), as follows:

```
$ NETCU [command]
```

The commands are:

- ADD ACCESS_LIST

- ADD SERVICE
- MODIFY SERVICE
- REMOVE ACCESS_LIST
- SHOW ACCESS_LISTS

See the *NETCU Command Reference* for details on each command.

You first set up an access list using the ADD ACCESS_LIST command to specify the hosts or groups of hosts you want to have explicit use of the service. Access lists should be defined in SERVERS.COM. You then define (or modify) a service using the ADD SERVICE (or MODIFY SERVICE) command and use the /ACCESS_LIST qualifier to indicate the access list number. If a host's internet address is not in the permitted list, it cannot use the service. Any connections the host makes to that service are immediately closed. The special /ACCESS_LIST=0 entry means that all users are allowed access to that service.

An implicit DENY is added to the end of each access list, meaning that you need an explicit permit to allow access. Also, an empty access list denies all users access to the service.

Subnet Masks

Consider the ADD ACCESS_LIST command syntax:

```
$ NETCU ADD ACCESS_LIST list condition ia mask
```

The *list* is the access list number, *condition* is either DENY or PERMIT, *ia* is the internet address, and *mask* is the subnet mask. The *mask* is the most significant field used to determine the order of entries in the file, followed by DENY before PERMIT conditions, followed by the address.

Often in entries without masks, all DENY entries are listed before all PERMIT entries. If you do use masks, this order can change based on your intent. Consider the following sequence of commands where you want to deny access to a Class C network yet permit access for a particular host on that network:

```
$ NETCU ADD ACCESS_LIST 1 DENY   192.168.142.0  255.255.255.0 -
_$ /MESSAGE="Sorry, access denied."
$ NETCU ADD ACCESS_LIST 1 PERMIT 192.168.142.25
$ SHOW ACCESS_LISTS
List Condition Internet Address Address Mask    Access Denied Message
---- --------- ---------------- ------------    ---------------------
1    PERMIT    192.168.142.25   255.255.255.255 "Sorry, access denied."
     DENY      192.168.142.0    255.255.255.0
```

The longer mask (255.255.255.255) is more specific about the entry and has priority in the file. That is why the PERMIT entry appears before the DENY entry, so that host 192.168.142.25 has access while other hosts on its Class C network do not.

Note that the `Access Denied Message` you can add using the /MESSAGE qualifier applies to DENY entries for a particular list. The message appears next to the first list entry in a SHOW ACCESS_LISTS display, even if it is a PERMIT entry as in the previous example.

Examples

The following example restricts FTP server access to hosts only on the local network.

```
$ NETCU ADD ACCESS_LIST 21 PERMIT local-network local-network-mask
$ NETCU MODIFY SERVICE 21 TCP/ACCESS_LIST=21
```

The following example uses an empty access list to make a service temporarily unavailable:

```
$ NETCU REMOVE ACCESS_LIST 1
$ NETCU MODIFY SERVICE FTP TCP/ACCESS_LIST=1
```

To make the service available again later, use:

```
$ NETCU MODIFY SERVICE FTP TCP/ACCESS_LIST=0
```

The 0 access list is special and means all users are allowed access.

Outgoing Access Restrictions

To use outgoing access restrictions, you create a file that contains the list of restrictions. Once you create this file, load it using the SET OUTGOING_ACCESS_RESTRICTIONS command in the Network Control Utility (NETCU). Once loaded, TCPware checks the outgoing access restrictions list whenever an active, outgoing connection attempt is made. If the outgoing access restrictions list permits the access, the connection attempt is allowed. If not, the connection attempt fails with a privilege violation status.

When building an outgoing access restrictions list, be sure to add permit entries for all permitted activity, since an implicit "deny everything else" is always at the end of the list. Therefore, you need an explicit permit entry to allow access.

The format of outgoing access restrictions list entries is as follows:

action ID [destination [mask] [operator port]]

Table19-1 describes the fields in each entry.

Table 19-1   Fields in an Outgoing Access Restrictions File Entry

Field...	With valid values...	Description
action		Keyword or keyword list (separated by commas; no spaces in between). Do not use PERMIT together with DENY.
	permit	Permits access.
	deny	Denies access.
	log	Logs the access attempt in the NETCP.LOG file.
	alarm	Generates an OPCOM message on the access attempt.
ID	username or UIC	Local user's username or UIC, such as `SMITH,` `[10,100],` or `[SMITH]`.
	rights-ID	Local user's rights identifier, such as ACCOUNTING.
	*	Wildcard used by itself (to mean "anyone") or as in `[*,100]` or `[10,*]`.
destination	Example: 192.168.95.126	Destination IP address to check, in dotted-decimal format. If omitted, indicates all destination addresses.
mask	Example: 255.255.255.0 (This example sets only the first three bytes in the destination address as significant. This restricts access to a class C network.)	Destination address mask to check, in standard bit mask format. Each bit set indicates that the corresponding bit in the destination field is significant. If omitted, TCPware determines the mask based on whether the host portion is 0; if 0, TCPware uses the network or subnet mask; if not 0, TCPware uses the mask 255.255.255.255 (meaning all bits are significant).
operator	lt le eq ge gt	Use with the port field: Less than Less than or equal to Equal to Greater than or equal to Greater than
port	port-number port-name	Destination port specification of either the port number (such as 2049 for the NFS port) or the port name (such as SMTP). If a name, must be defined it in the TCPWARE:SERVICES. file.

The order of the entries in the file is important. TCPware looks at each condition in sequence and stops testing after the first match that involves a permit or deny, ignoring any subsequent condition for an address. To use outgoing access restrictions:

1	Create outgoing access restrictions list entries in a file.
2	Apply the file entries to your system using commands in NETCU.

To create an outgoing access restriction file, edit a file and add entries in the format described. For example, you can set the following outgoing access criteria:

```
! Log all connections
log * 0.0.0.0 0.0.0.0
! Deny access to SMTP applications over the network
deny * 0.0.0.0 0.0.0.0 eq 25
! Permit all connections to the INTERNET_USER rights identifier only
permit internet_user 0.0.0.0 0.0.0.0
```

Any number of spaces or tabs can separate each entity. Lines beginning with an exclamation point (**!**) are comment lines. Use the dash (**-**) continuation character at the end of a line continuing onto the next.

An implicit **deny *** (deny all) is the last entry in the file. This means that an explicit **permit** entry must exist to allow any outgoing access. However, **permit** access always exists for privileged users (with SYSPRV or BYPASS privilege or running from a system UIC).

In addition, facilities that use a TCPware internal interface are also considered privileged users; for example, any application that uses the PWIPDRIVER interface (such as the PATHWORKS V5 server and the DECnet/OSI product from Compaq).

Once you load the file using SET OUTGOING_ACCESS_RESTRICTIONS, the outgoing access restrictions indicated in the file go into effect. You can use a subsequent SHOW OUTGOING_ACCESS_RESTRICTIONS command to display the results:

```
$ NETCU SET OUTGOING_ACCESS_RESTRICTIONS HOMERRESTRICT.DAT
$ NETCU SHOW OUTGOING_ACCESS_RESTRICTIONS
TCPware(R) for OpenVMS Outgoing Access Restrictions List
Actions  Userid        Destination Address Destination Mask  Port
-------  ------        ------------------- ----------------  ----
LOG      *             0.0.0.0             0.0.0.0
DENY     *             0.0.0.0             0.0.0.0           EQ 25
PERMIT   INTERNET_USER 0.0.0.0             0.0.0.0
```

Setting and Showing

Use the outgoing access restrictions commands on the NETCU level. The commands are:

- SET OUTGOING_ACCESS_RESTRICTIONS
- SET NOOUTGOING_ACCESS_RESTRICTIONS

- SHOW OUTGOING_ACCESS_RESTRICTIONS

All commands require OPER privilege. The SET OUTGOING_ACCESS_RESTRICTIONS command only affects the running system. To load the outgoing access restrictions each time you start TCPware, you can:

1	Place outgoing access restrictions in the TCPWARE:TCPWARE_OUTGOING_RESTRICTIONS.DAT file (STARTNET loads this file if it exists).
2	Add a SET OUTGOING_ACCESS_RESTRICTIONS command to the TCPWARE:ROUTING.COM file.

See the *NETCU Command Reference* for details about these commands.

CAUTION! Avoid using the SET OUTGOING_ACCESS_RESTRICTIONS NLA0: command. You can deny all outgoing access.

Examples

Examples of how you might use outgoing access restrictions include:

- For service bureaus or sites that want to limit which users are allowed access to TCP-based services, you might consider creating an INTERNET_USER rights identifier, and granting it to those users that purchase or need TCP/IP access. Then, you would create a very simple outgoing access restriction list such as:

  ```
  permit INTERNET_USER
  ```

  This list would permit all users with the INTERNET_USER rights identifier to establish TCP connections. Anyone without that rights identifier would not be allowed to establish TCP connections.

  Note that users can still send SMTP mail using SMTP-OpenVMS mailer. To disable use of this service, you would need to add ACLs to the TCPWARE:SMTP_MAILSHR.EXE file to allow only users with the INTERNET_USER rights identifier access.

- If you have problems with TELNET users accessing the SMTP port and creating fake mail messages, prevent this by using an outgoing access restriction list:

  ```
  deny * 0 0 eq smtp
  permit *
  ```

  Note that users can still send and receive SMTP mail using SMTP-OpenVMS (which runs with privileges that allow it to bypass the outgoing access restrictions list).

- If you want to log all outgoing connections:

  ```
  permit,log *
  ```

  To alarm all connections, use:

  ```
  permit,alarm *
  ```

  You can also log or alarm selected outgoing connections. For example, to log all outgoing SMTP connections:

```
permit,log * 0 0 eq smtp
permit *
```

Note that the second entry (**`permit *`**) permits all other outgoing connections.

Chapter 20

Packet Filtering

Introduction

This chapter describes how to manage TCPware packet filtering.

Using Packet Filtering

Packet filtering is used today in almost all (from basic to sophisticated) security *firewalls*. Packet filtering firewalls apply filtering rules to each packet received to determine whether to accept or discard it. These filtering rules specify the protocol, source and destination IP addresses, and destination ports (for TCP and UDP) for accepted or discarded packets.

You use packet filtering on routers between an internal network and one or more external networks (such as a connection to the Internet). Packet filter rules restrict what may come in over the interface connected to the external network.

Packet filtering can also be useful on hosts. For example, you can restrict the hosts allowed access to services. In particular, these are UDP-based services and services that the TCPware master server does not activate, and thus cannot use incoming access restrictions.

TCPware's packet filtering support is similar to what many routers provide. Once you create a file containing the packet filter rules and load it for an interface, any IP datagrams received on that interface are filtered before being processed or forwarded. (Packets are only forwarded if you enable forwarding; see the NETCU ENABLE FORWARDING command for details.)

Note that when you use packet filtering, each and every datagram received on the interface is filtered. This increases processing overhead depending on the size of the filter list.

Packet filtering can be an effective and useful security mechanism. However, it cannot solve all of your security problems. To be effective, you must construct the filtering rules carefully.

Cautions

Consider the following cautions when setting up packet filtering on an interface:

- Packet filtering does not use state information. Each datagram is filtered without any knowledge of packets that preceded it.

  This means that for UDP-based applications, it is not possible to add a rule that says to accept replies to requests. This also affects connection-oriented protocols such as FTP that use two connections, one for commands and the other for data.
- Fragmented datagrams for UDP or TCP are difficult to filter, since only the first fragment has the necessary port information. TCPware solves this problem by applying the filter rules only to the first fragment of UDP and TCP datagrams. The other fragments are accepted and processed or forwarded, but are eventually discarded since they cannot be reassembled without the first fragment. For all other IP protocols, the filter rules apply to each fragment.
- To set up secure packet filtering lists, you need a detailed knowledge of IP, ICMP, TCP, UDP and applications protocols.

Suggested reading includes the protocol RFCs (listed elsewhere in the TCPware documentation) and books such as Cheswick, William R. & Steven M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker.*

Packet Filter File

Packet filtering uses a filter list to determine whether you can receive a datagram. Filter lists are in packet filter files having the .DAT extension by default. Create one of these files first and then edit it. Format each file entry in the manner described in Table 20-1.

Table 20-1   Fields in a Packet Filter File Entry

Field...	With valid values...	Description
Entry Format: ***`action protocol saddr smask [sport] daddr dmask [dport [doption]]`***		
`action`	**`permit`** **`drop`** **`deny`**	**`permit`** permits the datagram; **`deny`** denies the datagram and sends the ICMP; **`drop`** denies the datagram without sending an ICMP destination unreachable message to the sender.
`protocol`	**`ip`** ***`ip-number`*** **`tcp`** **`udp`** **`icmp`**	Protocol to check: the values indicated or the numeric IP protocol number. The value **`ip`** matches any IP protocol.

Table 20-1 Fields in a Packet Filter File Entry (Continued)

Field...	With valid values...	Description
saddr	Example: `192.168.123.123`	Source IP address to check.
smask	Example: `255.255.255.255`	Source address mask to check, in standard bit mask format. To match a single address, use 255.255.255.255. To match any address, use 0.0.0.0
sport	***operator*** ***operand*** **lt** ***port*** **le** **eq** **ge** **gt** **ne**	Optional source port specification to check (for TCP and UDP entries only). Consists of an operator, space, and port name or number. If port name, must be defined in the `TCPWARE:SERVICES.` file. If omitted, any source port is valid. Example: **gt 1023**
daddr	Example: `192.168.123.123`	Destination IP address to check.
dmask	Example: `255.255.255.255`	Destination address mask to check, in standard bit mask format, as in *`s-mask`* above.
dport	***operator*** ***operand*** **lt** ***port*** **le** ***icmp-msg-type*** **eq** **ge** **gt** **ne**	Optional destination port (for TCP and UDP entries) or ICMP message type specification consisting of an operator, space, and operand. If a port name, must be in the TCPWARE:SERVICES. file. If *icmp-msg-type*, use: 0 Echo Reply 3 Destination Unreachable 4 Source Quench 5 Redirect 8 Echo 11 Time Exceeded 12 Parameter Problem 13 Timestamp 14 Timestamp Reply 15 Information Request 16 Information Reply

Table 20-1   Fields in a Packet Filter File Entry (Continued)

Field...	With valid values...	Description
`doption`	**`established`**	Matches only established connections (TCP segments with ACK or RST bits set).

Each entry specifies a packet filtering condition for a particular protocol type, source or destination address and mask, and destination port and mask specification, with certain additional options. The system looks at each condition in sequence, looks for a match, and takes a permit (accept) or deny (reject) action. The system stops testing conditions after the first match. This means that the order of the entries in the file is important; if the file lists a subsequent condition for an address, the system ignores it.

An implicit deny terminates the list of entries in the packet filter file. This means that if no condition matches, the system rejects the datagram. To use packet filtering:

1	Create address list entries in the packet filter file.
2	Apply the list to interfaces on your system by using packet filtering commands.

To create a packet filter file, edit a file and add address list entries in the format described.

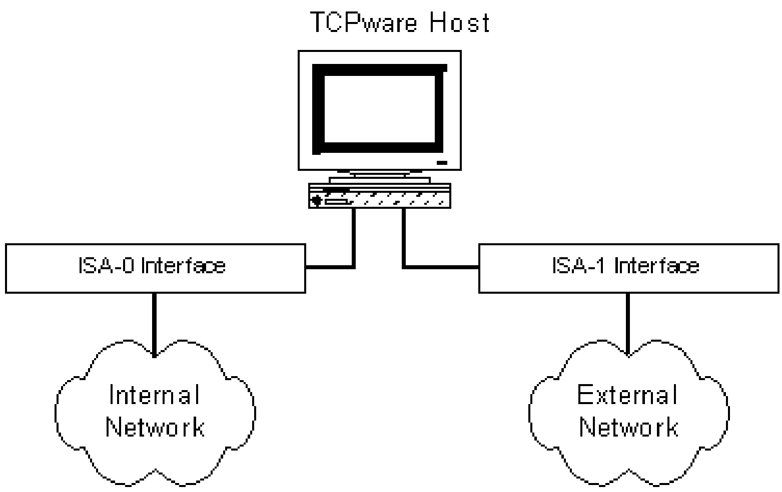
Any number of spaces or tabs can separate each entity. Lines beginning with an exclamation point (**`!`**) are comment lines. You can use the dash (–) continuation character at the end of a line that needs to continue onto the next.

To apply the list to a particular network interface or interfaces on your system, use the SET FILTER command, as described in *Setting and Showing*.

Configuration Recommendations

A packet filtering configuration might look something like Figure 20-1. A packet filter is installed at the ISA-1 interface to prevent unwanted packets from the external networks from entering the internal network. TCPware's packet filters are applied to packets received on a specific interface (no filtering of transmitted packets is done).

Figure 20-1 Tunneling DECnet over IP



Constructing the address filter list requires care in that you only want to let in the packets you need. Here are some recommendations in setting up an address filter list for an interface:

- Add an entry to prevent IP spoofing (that is, having an external host send a datagram as if it came from a local machine). For a router (as in Figure 20-1), this makes sense since no datagram received from an external network should ever have a local source address. Add the following entry to the filter list for the external interface (ISA-1 in Figure 20-1):
  deny ip *local-network local-network-mask*
- Be careful with services that use "unprotected" port numbers (greater than 1024). Some examples are NFS (port 2049) and X Windows (port 6000 and higher). Explicitly denying these services is a good idea:
  ```
  deny udp 0 0 0 0 eq 2049
  deny tcp 0 0 0 0 eq 2049
  deny tcp 0 0 0 0 eq 6000
  deny tcp 0 0 0 0 eq 6001
  ```
- Prevent broadcast and loopback packets from entering your network. It is best to restrict the broadcast (the first two of the following entries) to an external interface; apply the loopback restriction (the last entry) to any interface:
  ```
  drop ip 0.0.0.0 255.255.255.255
  drop ip 255.255.255.255 255.255.255.255
  drop ip 127.0.0.0 255.0.0.0
  ```
- Guard against datagrams from invalid source addresses when connected to the Internet (provided you are not using these network numbers for internal-only traffic purposes). Add the following to the filter list for the external interface (ISA-1 in Figure 20-1):
  ```
  drop ip 10.0.0.0 255.0.0.0
  drop ip 172.16.0.0 255.240.0.0
  drop ip 192.168.0.0 255.255.0.0
  ```

- You generally need to allow in domain name (DNS) requests using:
  `permit udp 0 0 0 0 eq 53`

  Whether to allow TCP DNS traffic (usually used for zone transfers) is also something to consider. To disallow TCP DNS traffic, add:

  `deny tcp 0 0 0 0 eq 53`
- You should not be concerned with what services local users use in the external world. You would want to add:
  `permit tcp 0 0 0 0 gt 1023 established`

  This allows all TCP datagrams in to ports greater than 1023 that have either the ACK or RST bits set in the TCP flags. Connection establishment requests have just the SYN bit set, so they are not allowed in by this entry.

  You may want to drop the **`established`** option if you want to allow incoming connections to unprotected ports. This would allow use of the FTP PASV capability.
- You may offer services to the external world such as a World Wide Web or anonymous FTP server. Add the following entries:

```
permit tcp 0 0 (web-server) 255.255.255.255 eq 80
permit tcp 0 0 (ftp-server) 255.255.255.255 eq 21
```

  If you have several hosts for each service, add an entry for each. Note that for the FTP Server, the data connections are normally outgoing and thus the earlier **`permit tcp 0 0 0 0 gt 1023 established`** configuration works to allow these. However, if users switch to PASV mode, the connections will be incoming (to unprotected port numbers) and therefore the **`permit tcp 0 0 0 0 gt 1023`** configuration (without the **`established`** option) may be more effective.
- Allow all ICMPs except ICMP redirects in:

```
deny icmp 0 0 0 0 eq 5
permit icmp
```

  This is useful for informing hosts about problems. But it can open up denial of service attacks, especially if hosts are not careful about the ICMP redirects that they accept. That is why discarding them is recommended.
- Watch the order of the entries in the table carefully.

```
permit tcp 0 0 0 0 gt 1023
deny tcp 0 0 0 0 eq 2049
```

  This entry would not work since the permit entry allows the datagram in and processing stops as soon as a match is found. TCPware processes the entries in the order in which you specify them.
- Remember that an implicit "deny everything" is added to the end of the filtering list. This means that to permit a datagram in, you need to have a permit entry in the list.
- Once you applied your filter list, test it first. Get an account on a host on an outside network that you can use to connect to your local hosts. Check that you are not allowing any access you do not want, and that you are allowing access that you do want. If something is not right, modify the filter list, reload it, and retest.

While packet filtering is very useful, it is by no means the only step you should take to secure your network. You must take special care to secure the system to assure that it cannot be compromised.

One way to do this is to greatly limit the services it offers.

Setting and Showing

The commands to set and show packet filters are:

SET FILTER	Associates a packet filter list with particular network lines
SET NOFILTER	Removes a previously associated packet filter list from network line(s)
SHOW FILTER	Displays the current packet filter list for network lines

See the *NETCU Command Reference* manual for details on these commands.

Setting at Startup

When you start TCPware, the STARTNET procedure looks for a TCPWARE:FILTER-*line-id*.DAT file for each interface it starts. If the file exists, STARTNET issues the following NETCU command to set the filter list for the interface:

```
SET FILTER line-id TCPWARE:FILTER-line-id.DAT
```

You can also add the necessary NETCU commands to the TCPWARE:ROUTING.COM file.

Chapter 21

Managing Token Authentication

Introduction

This chapter describes how to manage the TCPware Access Control Encryption (ACE) Client (ACE/Client) and how to set up user accounts in the Security Dynamics ACE/Server.

ACE/Client

The TCPware Token Authentication system consists of Security Dynamics Corporation's Access Control Encryption Server (ACE/Server) authentication software running in a trusted environment and TCPware ACE/Client software on devices that are to be protected by SecurID authentication. A backup server that provides backup authentication services if the main server is not running, administration software, and audit trails are part of the server environment.

Security Dynamics' SecurID smart card and ACE/Server software produce a two-factor authentication process that can provide an effective, secure environment. The SecurID smart card contains a microprocessor that generates and displays a new, unpredictable password (card code) every 60 seconds. The card displays this unique password, which is different for each card, on a liquid crystal display. Each card is programmed with a unique seed number and Security Dynamics' powerful proprietary algorithm.

The following TCPware components and OpenVMS functionality use the TCPware ACE/Client to authenticate a user before allowing access:

- FTP-OpenVMS
- RLOGIN command
- TELNET-OpenVMS
- SET HOST command

The TCPware ACE/Client is designed to do the following:

- Monitor and control token-holders
- Authenticate for interactive logins

- Authenticate based on network logins
- Authenticate based on remote logins

Terms

Special terms used in this chapter include:

ACE/Client	Computer or other device that is protected by the ACE/Server to prevent unauthorized access. Designated users of this computer must provide a valid SecurID PASSCODE in order to log in.
ACE/Server	Security Dynamics Corporation's authentication software running on a TCP/IP networked UNIX machine, providing authentication, administration, and audit trail services.
CERT	Computer Emergency Response Team.
PASSCODE	User's PIN plus the tokencode displayed in the SecurID token.
PIN	User's Personal Identification Number. The PIN is one factor in the Security Dynamics authentication system.
`sdadmin`	Program used on the main server to administer the ACE/Server system.
`sdconf.rec`	ACE/Server configuration file that contains information about the server that is distributed to the ACE/Clients. The ACE/Clients must have this file on their system in order to determine what parameters are needed to establish communication with the server.
SDI Encryption	Security Dynamics' proprietary encryption.
`securid.`	Node secret file that contains a random string only known between the client and server. Along with other information, `securid.` is used to encrypt and decrypt messages between the client and server.
token	Device that displays a SecurID tokencode. It may be a standard SecurID Card, a SecurID Key Fob, or a SecurID PINPAD card. The token is one of the factors in SecurID two-factor authentication. The other factor is the user's PIN.
two-factor authentication	Authentication method used by the ACE/Server in which the user must enter a secret, memorized personal identification number (PIN) and the current code generated by the user's assigned SecurID token. The PIN and tokencode make up the PASSCODE.

Documents and Standards

See the following documents for more information:

ACE/Server V2.2 Administration Manual, August 1996
ACE/Client for OpenVMS V1.0, December 1995
ACE/Server V2.2 Installation Guide, August 1996
Developers' Guides to the ACE/Server V2.2 Database and Client API, August 1996
ACE/Server Client Implementation Guide, August 26 1996.
Security Dynamics SecurID FAQ, `http://www.securid.com/service/FAQs/`

ACE/Server Functions

To administer the ACE/Server functions, use the `sdadmin` program, which you can run either in graphical interface mode or in character mode. To use graphical interface mode, go to the proper directory location of the program on the server machine, and enter the following:

```
# directory-path/sdadmin
```

To use character mode, enter the following:

```
# directory-path/sdadmin -c
```

Note! See Security Dynamics' *ACE/Server Administration Manual* for full details on server administration. The following is only a review of the functions.

The following is a sequence of steps on the server:

Step	Action	Description
1	**Add token records to a new ACE/Server database**	From the **`Token`** pull-down menu, select **`Import Tokens`**, enter the path and filename of the token records file, and select **`List Tokens`** to verify.
2	**Create clients**	From the **`Client`** pulldown menu, select **`Add Client`** and add a client. To duplicate information activation information from one client to another, use the **`Copy Client`** selection. Repeat this as often as needed.
3	**Create groups**	From the **`Group`** pulldown menu, select **`Add Group`** and add a group (this is optional but useful) along with its members (although you can do this later). To duplicate information activation information from one group to another, use the **`Copy Group`** selection. Repeat this as often as needed.

4	**Set system PIN parameters**	From the **`System`** pulldown menu, select **`System Parameters`** and set the PIN parameters such as whether to allow alphanumeric or just numeric PINs, if they should all be the same character length, and whether the system generates them or the user can generate them.
5	**Create users and assign tokens to them**	From the **`User`** pulldown menu, select one of the following: **`Add User`**, **`Edit User`**, or **`Copy User`** to create users (this is also where you can determine group membership).

The server administrator's responsibilities also includes distributing tokens to users and educating them about their security responsibilities (such as keeping PINs secret and securing and protecting their tokens).

The server administrator can also monitor the authentication process on the server and generate reports using `sdadmin`. Do this from the **`Report`** pulldown menu and by selecting **`Activity`**. Enter the parameters for the report and select **`OK`**. You can also log statistics.

ACE/Client Logicals

The logicals listed in Table 21-1 define the ACE/Client environment.

Table 21-1 ACE/Client Logicals

Logical	**Description**
TCPWARE_ACECLIENT_CL	Points to the shareable image activated by LOGINOUT when login is performed.
TCPWARE_ACECLIENT_ DATA_DIRECTORY	Points to the directory that contains ACE/Client data files. Set by the `Enter directory were the TCPware ACE/Client data file resides:` prompt in CNFNET.
TCPWARE_ACECLIENT_ENABLE	If set to 1, indicates that authentication by the TCPware ACE/Client is enabled. Set by the `Do you want to use the TCPware ACE/ CLIENT to authenticate user login?:` prompt in CNFNET.
TCPWARE_ACECLIENT_ NETWORK	If set to 1, indicates that authentication is performed on logins over network terminals, for example, _NT physical devices created if using TELNET. Set by the `Do you want to authenticate user network logins?` prompt in CNFNET.

Table 21-1 ACE/Client Logicals (Continued)

Logical	Description
TCPWARE_ACECLIENT_ PASSCODE_TIME	Number of seconds allowed for the user to input the PASSCODE. Set by the `Enter the PASSCODE input timeout time:` prompt in CNFNET.
TCPWARE_ACECLIENT_REMO	If set to 1, indicates that authentication is performed on logins over remote terminals, for example, _RT physical devices are created if using SET HOST. Set by the `Do you want to authenticate user remote logins?:` prompt in CNFNET.
TCPWARE_ACECLIENT_SHR	Points to the ACE/Client API.

Disabling ACE/Client

If there is a malfunction with the TCPware ACE/Client or if the main ACE/Server and the backup ACE/Server are not running and the TCPware ACE/Client is enabled, users who are designated for Token Authentication are not allowed access to the system. To disable the TCPware ACE/Client, perform the following command:

```
$ @TCPWARE:SHUTNET ACECLIENT
```

If you wish to disable the TCPware ACE/Client but cannot log in the system to do so, use the console to login. The console should be in a secure area and login is granted if the user password is correct. It does not prompt you for the `PASSCODE:`. Once you are logged in, you can disable the TCPware ACE/Client with the command described.

If you do not have a console, shut down the system and perform a minimum reboot. Log in and run CNFNET to disable the TCPware ACE/CLIENT for your particular configuration by responding with NO to the following prompt:

```
Do you want to use the TCPware ACE/CLIENT to authenticate user login ? NO
```

In the event that you cannot disable the TCPware ACE/Client using SHUTNET, you can do the following to manually disable it:

- Check the system, executive logical LGI$CALLOUTS to make sure that it does not include the TCPWARE_ACECLIENT_CL logical in its definition. If it does, redefine LGI$CALLOUTS to exclude TCPWARE_ACECLIENT_CL.
- Check the SYSGEN parameter LGI_CALLOUTS to make sure that it represents the correct number of shareable images as shown in the system, executive logical LGI$CALLOUTS. If it does not, set it to the correct value as follows in SYSGEN:

```
SYSGEN> USE ACTIVE
SYSGEN> SET LGI_CALLOUTS number-shareable-images
SYSGEN> WRITE ACTIVE
SYSGEN> EXIT
```

However, if you start TCPware as part of your site auto-startup, it enables the TCPware ACE/ Client after rebooting. Use CNFNET to permanently disable the TCPware ACE/Client.

Database Transfer and Startup

To use the ACE/Client, you must copy the ACE/Server configuration file, SDCONF.REC, from the ACE/Server to the ACE/Client machine. It must go into the TCPWARE_ACECLIENT_DATA_DIRECTORY described in the previous section.

If the TCPware ACE/Client and the ACE/Server do not have the same SDCONF.REC file, communication between them will be impossible. To view the SDCONF.REC file to make sure it matches the one one on the server, run the TCPWARE:ACEMAIN_CL.EXE utility on the client, as follows:

```
$ RUN TCPWARE:ACEMAIN_CL
```

Here is sample output:

```
Configuration file is version 6
Maximum number of ACE/Servers is 2
Maximum number of ACE/Client retries is 3
ACE/Client timeout is 10 seconds
DES has been disabled
Duress mode has been disabled
The ACE/Server is a trusted server
Number of bad Tokencodes allowed is 3
Number of bad PIN allowed is 3
ACE/Server service name is securid
Master ACE/Server protocol is udp
Master ACE/Server port 5500
Master ACE/Server is fred
Master ACE/Server address is 192.168.142.63
Slave ACE/Server is wilma
Slave ACE/Server address is 192.168.95.82
Slave ACE/Server port is 5510
```

If the display contains garbage characters, the SDCONF.REC file is corrupted. If the file is corrupted or the entries do not match those of the SDCONF.REC file on the ACE/Server, recopy the server file using FTP in binary mode.

The TCPware startup process:

- Checks to see if the ACE/Client is enabled.

- Defines the logicals described in the previous section.
- Installs the TCPWARE_ACECLIENT_SHR.EXE and TCPWARE_ACECLIENT_CL.EXE shareables.
- Defines the LGI$LOGINOUT_CALLOUTS to include TCPWARE_ACECLIENT_CL.
- Increments the SYSGEN parameter LGI_CALLOUTS by one.

Commands

The following NETCU commands are available to monitor and control which users should be authenticated before granting access:

ADD ACE_USER *username*	Adds the username to the ACE/Client database. The ACE/Server authenticates the user.
REMOVE ACE_USER *username*	Removes the username from the ACE/Client database. The ACE/Server no longer authenticates the user.
SHOW ACE_USER	Displays the ACE/Client database and lists the usernames that are being authenticated.
CREATE ACE_USER_DATABASE	Creates a new ACE/Client database and renames the old database if there was one.

ACE/Client Functions

The TCPware ACE/Client supports the following functions:

1	`Enter PASSCODE:` prompt to authenticate users by challenging them for SecurID PASSCODE information
2	New PIN operation
3	Next Tokencode operation
4	Backup servers
5	Encryption algorithm

The first three functions requires user interaction. Support of an optional backup ACE/Server is transparent to the users.

`Enter PASSCODE:` Prompt

This is the challenge for the SecurID authentication. Users must respond by entering their SecurID PASSCODE, comprised of their secret PIN, followed by the tokencode currently displayed on the user's SecurID token. PINPAD tokens require that the user's PIN be entered into the PINPAD token itself; the result that is displayed on the user's token is the complete PASSCODE and is entered as displayed. The PASSCODE prompt is normally displayed after the user has responded to the usual login prompts.

New PIN Operation

When a SecurID token is first assigned to a user, a PIN is not yet associated with it. A SecurID token cannot be used for authenticating until a PIN is assigned to it. When using their SecurID token for the first time, or in cases when a new PIN must be assigned to an existing user, users need to interact with the ACE/Server. This interaction is known as "New PIN mode."

Next Tokencode Mode

"Next Tokencode mode" requires that the user input a second successive tokencode from their SecurID Token. The ACE/Server puts a token into Next Tokencode mode if it has drifted out of synchronization with the server system's clock, or a PIN has been compromised and a hacker is attempting to guess a valid tokencode. Requiring two consecutive token-codes ensures that the user actually has possession of the SecurID token associated with the PIN that was entered. This feature must be supported for the ACE/Server to properly identify a user whose SecurID token clock may have drifted out of synch with the ACE/Server's clock. It is also required to allow the ACE/Server to perform evasive-action processing, as is the case when someone has learned a user's PIN and is attempting to guess the valid tokencode without having possession of the associated SecurID token.

Backup ACE/Server

The optional backup ACE/Server runs on a second UNIX machine and acts as a temporary backup to the ACE/Server. Backup ACE/Server software runs on any platform that can run the ACE/Server, but does not have to be the same operating system as that running the ACE/Server software. The backup ACE/Server is in regular communication with the ACE/Server via a dedicated TCP/IP socket. In the event of a failure of the ACE/Server platform or of the network connection, the backup ACE/Server processes authentication requests and generates audit trail records. The backup ACE/Server has all the features of the main ACE/Server except the administrative capabilities.

Encryption

The ACE/Server supports Security Dynamics' proprietary encryption. The ACE/Server system uses this method to secure transmissions between the ACE/Server, ACE/Client and backup ACE/Server.

Application Functionality

FTP, RLOGIN, TELNET, and SET HOST provide different login prompts for Token Authentication.

FTP

The user must first log in on a terminal session (such as TELNET, RLOGIN, or SET HOST) to obtain a new PIN if assigned for token authentication. The user cannot be in PIN mode (Next

Tokencode or New PIN mode) when in FTP, since there is no interaction between the FTP user and the ACE/Client.

When in FTP, the user must enter the username prompt and the PASSCODE at the usual **`Password:`** prompt. If the user is not assigned for token authentication, the user provides their password when FTP prompts for it.

TELNET, RLOGIN, and SET HOST

A user running TELNET, RLOGIN, or SET HOST must provide login information in order to be authenticated before access is allowed. In addition to the username and password information, the user is also asked to provide the PASSCODE.

User Messages

Following are the messages that the ACE/Client will display to the user who is attempting to authenticate.

- Enter PASSCODE:

  This is the ACE/Client prompt the user usually sees when attempting to authenticate. (New PIN mode, Next Tokencode mode and certain authentication failures are the exceptions). At this prompt, users must enter their SecurID PASSCODE. The PASSCODE is comprised of the secret PIN and the current tokencode displayed on the user’s SecurID token. The two values, when combined, are referred to as a PASSCODE. The format of the PASSCODE is dependent on the type of SecurID token being used.

- PASSCODE accepted

  This message is displayed on users’ screens when they enter a valid PASSCODE. The user is successfully authenticated and now has access to the ACE/Server-protected environment.

- Access denied

  The ACE/Client uses this message to indicate a failed authentication request (an invalid PASSCODE). The individual is denied access to the SecurID-protected system. After this message is displayed, the user may be prompted with another **`Enter PASSCODE:`** prompt. This message may be displayed for a number of reasons such as:

  – The user entered a valid PIN followed by an invalid tokencode. The entered value could be a previously used code or a guessed number.
  – When using a PINPAD token, the user entered an invalid tokencode.
  – The user entered an invalid PIN followed by a valid tokencode. The user entered a number other than the PIN associated with the token they are attempting to use.
  – When using a PINPAD token, the user entered an invalid PIN.
  – The user's SecurID token is disabled, either automatically to evade a system attack, or by administrative action.
  – A person attempting to gain unauthorized access is guessing PASSCODEs.
  – The user is not activated on the Client.
  – The Client was not found in the ACE/Server database.

- Access was attempted with a PASSCODE already in use.
- Mismatch of node secret or encryption type.
- The token, or the user's temporary access period, expired.

- Press **`Return`** to generate a new PIN and display it on screen

  or

  `Ctr/D` to cancel the New PIN procedure:

  This message is displayed when the user's SecurID token is in New PIN mode and the user entered the current tokencode (or tokencode and PIN). This prompt indicates that the ACE/Server is ready to generate a new PIN for the token and display it to the tokenholder, unless the process is aborted by pressing **`Ctr/D`**

- Enter your new PIN, containing 4 to 8 characters, or Press **`Return`** to generate a new PIN and display it on screen or **`Ctr/D`** to cancel the New PIN procedure:

  A message similar to this is displayed when the user's SecurID token is in New PIN mode and the user initiated a SecurID authentication. This prompt indicates that the ACE/Server is ready to generate a new PIN for the token or to allow users to create their own PIN.

- **`Enter your new PIN, containing 4 to 8 characters, or <Ctrl d> to cancel the New PIN procedure:`**

  This prompt appears when the user's SecurID token is in New PIN mode and the user initiated a SecurID authentication. In this case, the user MUST create a PIN or abort the operation using **`Ctr/D`**.The ACE/Server does not generate a PIN automatically. Pressing **`Return`** at this point re-displays the prompt. The new PIN, whether ACE/Server-generated or user-created, is displayed to the tokenholder, unless the process is aborted by pressing **`Ctr/D`**.

- PIN rejected

  The user selected an unacceptable PIN. The PIN specified by the user must conform to the system PIN specifications for length and allowable characters (digits only or letters and digits). PIN numbers must be composed of digits (0-9) or letters A-Z. PINPAD cards require that the PIN be composed exclusively of digits 0-9, the PIN cannot begin with zero, and the PIN length cannot exceed the length of the tokencode.

- **`Please enter the next code from your token`**:

  SecurID authentication is based on a patented time synchronization technology. If users have not used their SecurID token for an extended time, the SecurID token's clock may appear to the ACE/Server to be out of synch, or beyond the limits of normal synchronization. This is usually a result of mutual drift between the SecurID token's clock and the time source for the ACE/Server. In this event, the user will be prompted to enter the next tokencode.

Error Messages

Each of the following error messages indicates some failure with the ACE/Client, where *xxx* indicates the cause of the error:

ERRSDCONF	<Error reading *xxx*>
NOCLISERCOM	<Cannot initialize client-server communications>
FAILSYNCSER	<Failed to synchronize with server>
GTHOSTFAIL	<gethostname failed>
GTHOSTBYFAIL	<gethostbyname failed for host xxx>
ERRSELECT	<Error from select>
EXPSELECT	<Exception from select>
UNLOCUNIXFL	<Unable to locate *xxx* in the TCPware services file>
UNCRESOCK	<Unable to create ACE/Server socket>
UNBINDSOCK	<Unable to bind ACE/Server socket>
UNLOCSERHT	<Unable to locate ACE/Server host>
UNSNDSERV	<Unable to send to the ACE/Server>
ASSIGNFL	<SYS$ASSIGN failed: >
QIOWFAIL	<SYS$QIOW failed: >
CRENODESECR	<Can not create service file *xxx*>
`PASSCODETO`	`<PASSCODE Timeout, you have` *`xxx`* `seconds to input` the `PASSCODE>`

If the user is experiencing login failures, check the ACE/Server activity menu for additional information. Run `sdadmin` on the ACE/Server system and select the `Report` menu.

See the *ACE/Server Administration Manual* for details about authentication and ACE/Server errors.

Chapter 22

Managing Kerberos

Introduction

This chapter describes:

- Kerberos Server
- Kerberos management commands
- Kerberos administration server
- Kerberos `.KLOGIN` file
- Managing Kerberos for TELNET

Configuration Checklist

To set up Kerberos authentication between systems, all machines have to be configured for Kerberos: the Kerberos server, application server, and client application machines. You must first decide which machine in the Kerberos realm (domain) will be the Kerberos Server, and which machines will support application clients and servers with Kerberos authentication. You also need to define the Kerberos realm, the location of the user's Kerberos ticket file, and who the Kerberos users and administrators are going to be.

On the Kerberos Server machine, you need to create a Kerberos database, add application service entries to it, and create a service table file for each application server (such as the RLOGIN server machine). For each service table, you need to transfer (preferably hand-carry) a copy of the file to each application server. You also need to add user entries and administrator accounts (with access control list entries) on the Kerberos Server.

The service table file you create from the Kerberos database is *`host`*`-NEW-SRVTAB.` (where *`host`* is the specific host name) for each application server with Kerberos authentication. This file should be located within the same directory where the NETCU CREATE SRVTAB command was issued. The file must be copied (preferably hand-carried) from this directory to the application server's remote host and renamed **`TCPWARE:SRVTAB.`**. Also, Kerberos must be configured for each application server.

Only the Kerberos configuration needs to occur on each client application machine. This defines the Kerberos logicals needed. Kerberos must also be started on each machine. No Kerberos database entry is needed for client applications, but entries are needed for server applications and user accounts. See Figure 22-1.

Figure 22-1 Kerberos Configuration Checklist

- [ ] Decide which machine will be the Kerberos Server.
- [ ] Decide on the name of the Kerberos realm.
- [ ] Decide which Kerberos applications to support on each machine.
- [ ] Know which users will use the Kerberos applications.
- [ ] (Optional) Decide who will have a Kerberos administrator account, and with what access rights.

On the Kerberos Server machine:

- [ ] Configure and start Kerberos (@TCPWARE:CNFNET KERBEROS). At this point, Kerberos will stop until you complete the next few steps.
- [ ] Create the Kerberos database (NETCU CREATE KDB), specifying a master password.
- [ ] Stash the Kerberos master password (NETCU STASH MASTER_PASSWORD).
- [ ] Restart Kerberos (@TCPWARE:STARTNET KERBEROS).
- [ ] Add Kerberos database entries for each Kerberos user (NETCU ADD KDB).
- [ ] (Optional) Add a Kerberos administrator account (NETCU ADD KDB).
- [ ] (Optional) Add Kerberos Access Control List entries for the administrator account(s) (NETCU ADD KACL).
- [ ] Add Kerberos database entries for each application service and its machine (NETCU ADD KDB).
- [ ] Create service table (SRVTAB) files for each application service machine (NETCU CREATE SRVTAB).
- [ ] Handcarry the SRVTAB files to their respective application servers.

On each Kerberos application server machine:

- [ ] Make sure that the TCPWARE:SRVTAB. file is the newest one from the Kerberos database.
- [ ] Configure each system as a Kerberos application service (@TCPWARE:CNFNET KERBEROS).
- [ ] Start Kerberos on each system (@TCPWARE:STARTNET KERBEROS).

On the Kerberos application client machine:

- [ ] Configure each system as a Kerberos application service (no Kerberos application servers are needed during configuration).
- [ ] Start Kerberos on each system.

Server Concept

Kerberos provides network security by regulating user access to network services. Kerberos solves the problem of how a server can be sure of a client's identity by having both the client and server trust a third party, in this case the Kerberos Server. The Kerberos Server verifies the client's identity.

If you set up your machine as a Primary Kerberos Server, TCPware creates two processes:

- Kerberos Server (TCPWARE_KERBV4)

  The Kerberos Server handles authentication requests. This server uses UDP as its transport protocol.

- Kerberos Administration Server (TCPWARE_KADMV4)

  The Kerberos Administration Server handles ***remote*** administration of the Kerberos database (KDB). This server uses TCP as its transport protocol.

Configuring the Server

During configuration, CNFNET defines each of the logicals listed in Table 22-1.

Table 22-1   Kerberos Logicals

Logical	Description
TCPWARE_KERBV4_MAXAGE	Sets the maximum age of the Kerberos database
TCPWARE_KERBV4_PRIMARY	Sets the Primary Kerberos Server name
TCPWARE_KERBV4_REALM	Sets the realm name of the Kerberos Server
TCPWARE_KERBV4_RLOGIN	Determines if the RLOGIN Server mandates, accepts, or disallows any Kerberos request
TCPWARE_KERBV4_RSHELL	Determines if the RSH Server mandates, accepts, or disallows any Kerberos request
TCPWARE_KERBV4_SRVTYP	Sets the type of server (primary, or applications only)
TCPWARE_KERBV4_TELNET	Determines if the TELNET Server mandates, accepts, or disallows any Kerberos request
TCPWARE_KERBV4_TKFILE	Sets the location of the user's ticket file

Service Type

The TCPWARE_KERBV4_SRVTYP logical determines the type of service on the local machine. You can have the Primary Kerberos Server and Kerberos applications running on it, or just Kerberos applications running on it.

You usually set the service type during CNFNET. If you need to define the logical manually, do so in one of the following two ways:

1	$ **DEFINE/SYSTEM/EXEC TCPWARE_KERBV4_SRVTYP P** The **P** indicates that this machine is to have a Primary Kerberos Server and Kerberos applications running on it
2	$ **DEFINE/SYSTEM/EXEC TCPWARE_KERBV4_SRVTYP A** The **A** indicates that this machine is to have only Kerberos applications running on it.

Realm Name

The TCPWARE_KERBV4_REALM logical determines the name of the Kerberos realm. You normally set the name of the local realm during CNFNET. The following example shows how to define the logical manually:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_KERBV4_REALM DAISY.COM
```

DAISY.COM is the realm (domain) in which the Kerberos Server resides. To set up a realm:

- GateD or the TCPWARE:ROUTING.COM file (but not both) must be set up to handle other subnets within the Kerberos realm.
- Domain Name Services should be configured on both the client and server applications using Kerberos V4 authentication.

Primary Server Name

The TCPWARE_KERBV4_PRIMARY logical determines the name of the primary Kerberos server. All Kerberos functions use the primary server.

You normally set the name of the primary server during CNFNET. The following example shows how to define the logical manually with the primary server defined as CHARON:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_KERBV4_PRIMARY CHARON
```

Maximum Database Age

The TCPWARE_KERBV4_MAXAGE logical determines the maximum age (in seconds) of the Kerberos database. The Kerberos Server uses this logical to determine if the database is too old. If the logical value is -1, the Server does not check the database age. Otherwise, the value must be between 3600 (seconds, or one hour) and 259200 (seconds, or three days).

If, when checking the age of the database, the Server detects that it is too old, it logs an error message in the Kerberos log file (TCPWARE:KERBEROS.LOG). The Server then continues, using an "old" database, if needed.

You normally set the maximum database age during CNFNET. The following example shows how to define the logical manually with the maximum age of the database set to one hour:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_KERBV4_MAXAGE 3600
```

Management Commands

The management interface with Kerberos comprises the following NETCU commands:

ADD KDB	Adds an entry to the KDB
CREATE KDB	Creates the KDB
CREATE SRVTAB	Creates a Kerberos service table
DUMP KDB	Dumps the KDB into a text file
LOAD KDB	Loads the KDB from a text file
MODIFY KDB	Modifies an entry in the KDB
REMOVE KDB	Removes an entry from the KDB
SET MASTER_PASSWORD	Sets the Kerberos master password
SHOW KDB	Shows a KDB entry
STASH MASTER_PASSWORD	Stashes the master password

See the *NETCU Command Reference* for details.

All commands require the OPER or SYSPRV privilege and entry of the Kerberos master password.

NETCU prompts for the master password the first time you use a Kerberos management command, unless you specify /NOPROMPT with the command. In the latter case, TCPware reads the master password from the file created by the STASH MASTER_PASSWORD command. You do not need the master password for the remainder of the NETCU session.

Uppercase entries convert to lowercase in these commands. If you have mixed case values or want to preserve case, enclose the value in double quotes. Use an asterisk (`*`) as a wildcard in parameters, but do not combine it with any other character.

Kerberos Database

Use the CREATE KDB command to create the KDB and enter the master password. CREATE KDB is equivalent to the UNIX command `kdb_init`. The command format is:

```
CREATE KDB
Enter Kerberos master password:
Verifying, please re-enter:
```

You must enter the Kerberos master password at the prompt and re-enter it as verification at the next prompt. TCPware does not echo the password. However, keep the master password as secure as the password for the SYSTEM account.

The KDB, by default, is in the TCPWARE:PRINCIPAL.OK file. When you create the KDB, you can also specify an alternate location, using the /KDBFILE qualifier. CREATE KDB uses your local domain name by default as the Kerberos realm in the KDB. If you want to specify an alternate realm name, use the /REALM qualifier.

Stashing the Master Password

Use the STASH MASTER_PASSWORD command to stash the master password in a protected file (TCPWARE:KSTASH.KEY). This allows you to add entries to the KDB without being prompted for the password. STASH MASTER_PASSWORD is equivalent to the UNIX command `kstash`. The command format is:

```
STASH MASTER_PASSWORD
```

If you do not stash the Kerberos password, after you create the Kerberos database (using CREATE KDB) and try to use it, you will not get a response; the Kerberos Server terminates automatically when this happens. The Kerberos Server cannot get the master key.

To fix this, perform the following commands:

```
$ NETCU STASH MASTER_PASSWORD
$ @TCPWARE:SHUTNET KERBEROS
$ @TCPWARE:STARTNET KERBEROS
```

Adding Entries

Use the ADD KDB command to add an entry to the KDB. ADD KDB is equivalent to the UNIX command `kdb_edit`. The command format is:

ADD KDB *principal password* [*instance*]

where:

- *principal* is the client user's login name or the name of the service provided.
- *password* is the user password.

  Specifying **"RANDOM"** as the password generates a random password and is recommended for the application services entries; do not use this for Kerberos users or administrators. You can also specify **"NULL"** as the password; its entry will have no password.
- *instance* is usually omitted for a general Kerberos user, `admin` for an administrative user, or the name of the machine on which the Kerberos application resides.

You can specify the alternate location of the KDB (using the /KDBFILE qualifier). You can also indicate the maximum lifetime and expiration date of the KDB entry in minutes, using the /MAX_LIFE and /EXP_DATE qualifiers, respectively.

Modifying Entries

Use the MODIFY KDB command to modify an entry in the KDB. MODIFY KDB is equivalent to the UNIX command `kdb_edit`. The command format is:

MODIFY KDB *principal* [*instance*]

Entry of the master password and options are the same as for the ADD KDB command.

Removing Entries

Use the REMOVE KDB command to remove an entry in the KDB. REMOVE KDB is equivalent to the UNIX command `kdb_edit`. The command format is:

REMOVE KDB *principal* [*instance*]

You can also use /KDBFILE and /NOPROMPT with this command.

Showing Entries

Use the SHOW KDB command to show the status of KDB tickets. SHOW KDB is equivalent to the UNIX command `klist`. The command format is:

SHOW KDB *principal* [*instance*]

You can use /KDBFILE and /NOPROMPT with this command.

Changing the Master Password

Use the SET MASTER_PASSWORD command to change the master password you entered at the CREATE KDB `Enter Kerberos master password:` prompt.

SET MASTER_PASSWORD is equivalent to the UNIX command `kdb_util new_master_key`. The command format is:

SET MASTER_PASSWORD

As with creating the KDB, you need to enter the current Kerberos master password. You can then enter the new master password. You can also specify an alternate KDB file other than PRINCIPAL.OK using the /KDBFILE qualifier.

Dumping to Another File

Use the DUMP KDB command to dump the contents of the KDB into an ASCII text file. DUMP KDB is equivalent to the UNIX command `kdb_util dump`. The command format is:

DUMP KDB *filespec*

You can use /KDBFILE and /NOPROMPT with this command.

Loading from Another File

Use the LOAD KDB command to load the KDB from the ASCII text file produced by DUMP KDB. LOAD KDB is equivalent to the UNIX command `kdb_util load`. The command format is:

LOAD KDB *filespec*

You can use /KDBFILE and /NOPROMPT with this command.

Creating the Service Table File

Use the CREATE SRVTAB command to create an encrypted service table file for a remote host. The remote host's application servers use this to authenticate principals. CREATE SRVTAB is equivalent to the UNIX command `ext_srvtab`. The command format is:

CREATE SRVTAB *instance*

You can use the /KDBFILE, /NOPROMPT, and /REALM qualifiers with the SRVTAB command. The Kerberos Server uses the *instance* as part of the service table filename, as shown in the next example.

You must create a service table for the Kerberos *application server* after its service ticket entry is added to or modified in the Kerberos database. After adding a service ticket entry (using NETCU ADD KDB rcmd "RANDOM" BART) for remote host `bart`, create the service table for `bart` as shown in the next example:

CREATE SRVTAB BART

The Kerberos Server creates the `BART-NEW-SRVTAB.` file in the local directory. You must bring this file over to remote host `bart`. If `bart` runs TCPware, you must copy it to `bart's` TCPWARE:SRVTAB. file. If `bart` is UNIX, you must copy it into its `/etc/srvtab` file.

CAUTION! You must keep the service table file secure. Copy the file to a tape or disk for transfer to the remote host rather than transferring it over the network. This way, no one can intercept it.

Administration Server

The Kerberos Administration Server (KADMIN) allows remote administration of the Kerberos primary server database. Kerberos administrators can add, modify and show Kerberos user records. KADMIN also allows Kerberos users to change their Kerberos passwords. You can create Kerberos administrator accounts and the Access Control Lists (ACLs) used to control access to the database.

KADMIN automatically starts up when you configure the machine to be a Primary Kerberos Server. However, Kerberos users can only get access to KADMIN to change their Kerberos passwords, unless further configuration takes place.

Figure 22-2 shows a request to a KADMIN Server. The following list describes each step:

1	The KADMIN client sends a request to the Kerberos Server for a Kerberos administrator ticket.
2	The Kerberos Server checks the Kerberos database to see if the client is legitimate.
3	If the client is legitimate, the Kerberos Server returns a Kerberos administrator ticket to the client. (If the client is not legitimate, the Kerberos Server returns an error message.)

4	The KADMIN client places the ticket in a temporary ticket file. This ticket contains an encryption key.
5	The KADMIN client encrypts its request for service, opens a TCP connection, and sends the encrypted request to the KADMIN server for processing. The KADMIN server recognizes four operations: – A user changing his password – Changing a Kerberos user's password – A user displaying a Kerberos user record – Adding a Kerberos user record If the server detects an error at any point, it sends an error message to the KADMIN client.
6	The KADMIN server decrypts the request, and checks the ACL lists to see if the KADMIN client has the necessary privileges for the requested operation.
7	If the request is legitimate, the KADMIN server consults the Kerberos database, performs the requested operation, and saves the results.
8	The KADMIN server encrypts the status of the operation and any requested data, sends it back to the KADMIN client, and closes the TCP connection.

To perform this operation, use the following KADMIN commands at the NETCU level on the server system:

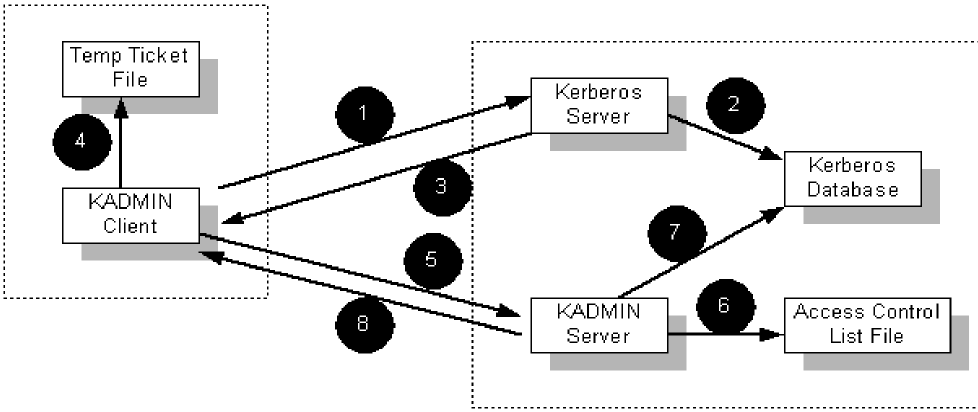
ADD KACL	Adds a Kerberos access control list (KACL) for the Kerberos database
REMOVE KACL	Removes a KACL entry
SHOW KACL	Shows the KACL entries

Use the following KADMIN commands at the NETCU level from a remote client system:

ADD KERBEROS USER	Adds a user entry to the KDB
MODIFY KERBEROS USER	Modifies a user record in the KDB
SET KERBEROS_PASSWORD	Changes a user's Kerberos password
SHOW KERBEROS USER	Lists a Kerberos user entry

See Chapter 2, *NETCU Commands*, in the *NETCU Command Reference* for details.

Figure 22-2 Kerberos Administration Server



Accounts

The first step in the KADMIN process is to create Kerberos administrator accounts by adding KADMIN users to the Kerberos primary server database. These accounts take the form:

***username*.admin**

For example, to create a Kerberos administrator account for `persephone`, use the following NETCU command on the Kerberos server, where **`persephone`** is the principal name, **`spring`** is the password, and **`admin`** is the instance that identifies the Kerberos administrator:

```
$ NETCU ADD KDB PERSEPHONE SPRING ADMIN
Enter Kerberos master password:
Verifying, please re-enter:
```

Use the following command to see the results of the operation:

```
$ NETCU SHOW KDB PERSEPHONE ADMIN
Name :              persephone
Instance :          admin
Expiration Date :   31-DEC-2099 23:59
Modification Date : 1-Feb-1999 09:21:09
Attributes :        0
Maximum Lifetime :  255
Key Version :       1
```

Access Control Lists

The next step is to add the ACL entries specifying what `persephone` can do. There are three possible operations: ADD, SHOW, and MODIFY access. ADD allows the administrator to add new Kerberos users. SHOW allows the administrator to show Kerberos user records. MODIFY allows the administrator to change the Kerberos password of other Kerberos users. Perform all these operations on the machine on which the Kerberos database resides.

To add ACL entries for Kerberos administrator account `persephone`, do the following at the NETCU prompt:

```
ADD KACL ADD PERSEPHONE ADMIN HADES.COM
ADD KACL MODIFY PERSEPHONE ADMIN HADES.COM
ADD KACL SHOW PERSEPHONE ADMIN HADES.COM
```

On the first line, the ADD after ADD KACL is the ACL type, **`persephone`** is the username, **`admin`** is the instance, and **`hades.com`** is the optional realm name. The above commands allow add, modify, and show operations.

To see the added ACL records, do the following at the NETCU prompt:

```
SHOW KACL PERSEPHONE ADMIN HADES.COM
ACL Type   Kerberos user
--------   -------------
ADD        persephone.admin@hades.com
MODIFY     persephone.admin@hades.com
SHOW       persephone.admin@hades.com
```

Examples

Once you set up the Kerberos administrator account, you can do the following remote administration operations through NETCU:

- Add a new Kerberos user, specifying the administrator password, as follows:
  ```
  ADD KERBEROS USER ACHILLES RUNNING /ADMIN=PERSEPHONE
  Administrator password for 'persephone':
  ```
- Show a Kerberos user record, specifying the administrator password, as follows:
  ```
  SHOW KERBEROS USER ACHILLES /ADMIN=PERSEPHONE
  Administrator password for 'persephone':
  Name :              achilles
  Instance :
  Expiration Date :   31-DEC-2099 23:59
  Attributes :        0
  Maximum Lifetime :  255
  ```
- Modify a Kerberos user record, specifying the administrator password, as follows:
  ```
  MODIFY KERBEROS USER ACHILLES FAST /ADMIN=PERSEPHONE
  Administrator password for 'persephone':
  ```
- Kerberos users can change their own passwords as follows:
  ```
  SET KERBEROS_PASSWORD ACHILLES
  Old password for 'achilles':
  New password for 'achilles':
  Verifying, please re-enter:
  ```

Kerberos for the Berkeley R Services

Establishing Kerberos authentication for RLOGIN, RSH, and RCP occurs during TCPware's CNFNET procedure.

See the *Installation & Configuration Guide*, Chapter 4, *Configuring the TCP/IP Services*, the *Configure Kerberos Applications* section.

In addition to setting up the Primary Kerberos Server, the configuration process implements an additional `klogin` service (if Kerberos authentication for the RLOGIN service is requested) and `kshell` service (if Kerberos authentication for the RSH or RCP service is requested). These services allow users to pass a Kerberos service ticket in place of a password to authenticate themselves.

The Berkeley R Commands use the .KLOGIN file to authenticate a remote Kerberos user. The `klogin` or `kshell` service checks the .KLOGIN file in the user's login directory.

Each line in the .KLOGIN file should contain a Kerberos principal in the following format:

name.instance@realm

For example:

```
fred@daisy.com
persephone.admin@rose.com
```

For details on Kerberos principals, see *Kerberos Services* in Chapter 18, *Managing Security*.

If the remote user is authenticated to one of the principals in the .KLOGIN file, the remote user has access to the service. The principal `username@local-realm` is granted access if there is no .KLOGIN file. Otherwise, standard authentication is used to log in to the server, if allowed.

Require, Allow, or Disable Requests

You can change whether you want to require, allow, or disable the RLOGIN or RSH server from handling Kerberos authentication requests. Do this by rerunning CNFNET KERBEROS and restarting the Kerberos Services.

You can select one of the following:

REQUIRED	Handle only Kerberos requests
ALLOWED	Handle both Kerberos and non-Kerberos requests
DISABLE	Reject all Kerberos requests

The RLOGIN and RSH servers use the value of the TCPWARE_KERBV4_RLOGIN and TCPWARE_KERBV4_RSHELL logicals, respectively, to determine how requests are handled.

Customizing the Kerberos Authentication Services

CNFNET implements the proper NETCU ADD SERVICE /ROUTINE=`create_rservice_kerberos` internal action routine during Kerberos configuration. This is used only in servicing RLOGIN or RSH (and RCP) Kerberos requests.

You can customize Kerberos authentication for RLOGIN and RSH after Kerberos configuration by using the NETCU ADD SERVICE command. If you customize the ADD SERVICE command, use the /ROUTINE=`create_rservice_kerberos` qualifier and value along with the `klogin` port and `stream` protocol (for RLOGIN service), or the `kshell` port and `stream` protocol (for the RSH service).

You can additionally show, modify, or remove Kerberos authentication for RLOGIN or RSH by using, respectively, the NETCU SHOW SERVICES, MODIFY SERVICE, or REMOVE SERVICE command. See the *NETCU Command Reference*.

Kerberos for TELNET

Kerberos configuration for TELNET occurs during TCPware's CNFNET procedure.

See the *Installation & Configuration Guide*, Chapter 4, *Configuring the TCP/IP Services*, the *Configure Kerberos Applications* section.

Note! You must also install and configure TCPware TELNET-OpenVMS for Kerberos authentication of TELNET requests to work. All the rules and commands for using TELNET-OpenVMS apply to TELNET with Kerberos authorization.

See Chapter 17, *TELNET-OpenVMS Management*, for details about TELNET logicals and options. See the *User's Guide*, Chapter 11, *TELNET: Connecting to Remote Terminals*, for details on using TELNET and a description of the available TELNET commands.

You can change whether you want to require, allow, or disable the TELNET server from handling Kerberos authentication requests. Do this by rerunning CNFNET KERBEROS and restarting the Kerberos Services.

You can select one of the following:

REQUIRED	Handle only Kerberos requests
ALLOWED	Handle both Kerberos and non-Kerberos requests
DISABLE	Reject all Kerberos requests

The TELNET server uses the value of the TCPWARE_KERBV4_TELNET logical to determine how requests are handled.

Chapter 23

IP Security Option

Introduction

This chapter describes how to manage the IP Security Option (IPSO) to protect IP datagrams over the network.

IPSO Security

IPSO is an option included in the header of a datagram that specifies the datagram's level of security. This option is a *label*. A system can check for the existence of an IPSO label in an incoming datagram and determine whether the datagram matches its specific security requirements. Likewise, a system can screen outbound datagrams based on their labels.

For an IPSO-supported system, if a datagram does not pass a label test, the system considers it out of range and rejects (discards) it. The system then sends an ICMP error message (if enabled) describing the condition to the datagram's originator.

The system manager must decide:

The consequences of using IPSO security	IPSO has a far reaching effect on all network applications running over IP.
The IPSO security option the system requires	The system manager can decide to implement the Basic Security Option or Extended Security Option.
The security level (or levels) for incoming or outgoing datagrams	Top Secret, Secret, Confidential, or Unclassified.
The protection authority (or combination of authorities) for incoming or outgoing datagrams	GENSER, SIOP-ESI, SCI, NSA, or DOE (for the meanings of these abbreviations, see *Security Levels and Protection Authorities*).

How to set IPSO on a system-wide and individual port (line) basis.	
What to do with unlabeled datagrams	The system manager can either reject such datagrams or implicitly label them so that they pass IPSO screening tests.
How to set security on ICMP error messages generated by out-of-range datagrams	In some cases, you must set IPSO on these messages so that the system does not reject them; in other cases, the system manager may want these messages to be "invisible."

Consequences

IPSO is for physically secure TCP/IP networks run by trusted system managers with the highest security clearance. You should segregate IPSO networks from non-IPSO networks. IPSO-protected datagrams should never enter a non-IPSO network.

IPSO affects everything in a TCP/IP-based network. For example, TELNET connections are impossible if receiving systems reject IPSO-labeled datagrams. This also affects FTP, NFS, FINGER, Line Printer Services, RSH, RCP, RLOGIN, among others.

IPSO also affects other protocols on which these applications may depend, such as the DNS, Kerberos, NTP, TIMED, RIP, and RAP. If a host cannot communicate with a DNS server, for example, all host information required to run an application becomes unavailable. In such cases, system managers must set all possible DNS servers with the same IPSO security.

Basic and Extended Security Options

IPSO incorporates a Basic Security Option and an Extended Security Option. Each IPSO-labeled datagram has in its IP header authorization information related to these options.

The Basic Security Option is well-documented and standardized, and includes the security levels and protection authorities described in the next subsection.

RFC 1108 minimally mentions the Extended Security Options, but they need more definition. Security managers can use extended options as custom, site-specific protocols they do not want to reveal. These options occupy extra space in the IP header after the basic option. You can only use them together with an existing basic option. TCPware supports the extended options in allowing or disallowing extended options in datagrams.

Security Levels and Protection Authorities

An IPSO security label consists of a classification level and a protection authority. The classification levels appear in Table 23-1 with their hexadecimal values. The protection authorities appear in Table 23-2 with their hex values and points of contact.

Table 23-1 IPSO Security Levels

Security level	Has hexadecimal value
Top_Secret	%X3D
Secret	%X5A
Confidential	%X96
Unclassified	%XAB

Table 23-2 IPSO Protection Authorities

Protection authority	Has hex value	With point of contact
GENSER	%X80	Designated Approving Authority per DOD 5200.28
SIOP-ESI	%X40	DoD Joint Chiefs of Staff
SCI	%X20	Director of Central Intelligence
NSA	%X10	National Security Agency
DOE	%X08	Department of Energy

Labeling as Opposed to Screening a Datagram

There are two distinctions in IPSO:

- Labeling an outgoing datagram
- Screening datagrams

Some datagrams already have IPSO labels when they reach your system, for example, by socket calls or other labeling methods. You can label "raw" datagrams by adding an implicit transmit label onto the datagram using a TCPware command. (You can also redefine already labeled datagrams in the same way.)

Screening a datagram involves checking whether to allow the labeled datagram in or out. You use TCPware IPSO commands for this purpose. You can screen on a system-wide and line-by-line basis, as discussed in the next section.

Commands describes the commands used to label and screen datagrams.

System and Line Basis Protection

You can set IP Security Options on a system-wide and line-by-line basis.

- The system requirements determine whether a datagram can enter or leave the system.
- The line requirements determine whether a datagram can enter or leave a specific port (line) on the system.

Datagrams destined for your system or the ones your system generates undergo system level IPSO checks. Datagrams just passing through (being forwarded) undergo only line checks on the port they use. All datagrams undergo line checks.

You can restrict types of security to certain lines or make security requirements for forwarded datagrams different than for datagrams destined for the system. You can also use system requirements to cover lines not specifically set. To be effective, system requirements should be supersets of all individual line requirements.

Both system and line settings consist of incoming and outgoing requirements. Example 23-1 shows how to use a SET IPSO /SYSTEM command so that both incoming and outgoing requirements are for a security level of UNCLASSIFIED and a protection authority of NONE. These are the default settings if you do not specify a level or authority. (Note that the levels show up as ranges of values with the SHOW IPSO command.)

Example 23-1 System Basis IPSO Setting

```
NETCU> SET IPSO /SYSTEM
NETCU> SHOW IPSO
Tcpware(R) for OpenVMS IPSO Configuration for System:

Label       Level                          Authorities
-----       -----                          -----------
In:         UNCLASSIFIED to UNCLASSIFIED   None
Out:        UNCLASSIFIED to UNCLASSIFIED   None


Implied
  Receive:  UNCLASSIFIED                   None
  Transmit: UNCLASSIFIED                   None
ICMP Error: UNCLASSIFIED                   None
```

To set the incoming and outgoing requirements separately, you could specify a command such as in Example 23-2. The incoming label now requires an UNCLASSIFIED to SECRET level range on a system basis. All the other values remain the same.

Example 23-2 System Basis IPSO Setting with Level Range Specified

```
NETCU> SET IPSO /SYSTEM /IN_LABEL=(LEVEL=(UNCLASSIFIED,SECRET))
NETCU> SHOW IPSO
Tcpware(R) for OpenVMS IPSO Configuration for System:

Label       Level                         Authorities
-----       -----                         -----------
In:         UNCLASSIFIED to SECRET        None
Out:        UNCLASSIFIED to UNCLASSIFIED  None


Implied
  Receive:  UNCLASSIFIED                  None
  Transmit: UNCLASSIFIED                  None
ICMP Error: UNCLASSIFIED                  None
```

If you want the line to accept just SECRET incoming datagrams, you could use a command such as in Example 23-3. Again, only the incoming line requirement changes.

Example 23-3 Line Basis IPSO Setting

```
NETCU> SET IPSO /LINE=SVA-0 /IN_LABEL=(LEVEL=SECRET
NETCU> SET IPSO /LINE=SVA-0 /RECEIVE_IMPLICIT_LABEL=(LEVEL=SECRET)
NETCU> SHOW IPSO
Tcpware(R) for OpenVMS IPSO Configuration for line SVA-0:

Label       Level                         Authorities
-----       -----                         -----------
In:         SECRET to SECRET              None
Out:        UNCLASSIFIED to UNCLASSIFIED  None


Implied
  Receive:  SECRET                        None
  Transmit: UNCLASSIFIED                  None
ICMP Error: UNCLASSIFIED                  None
```

You can also use SET NOIPSO to disable IPSO.

Unlabeled Datagrams

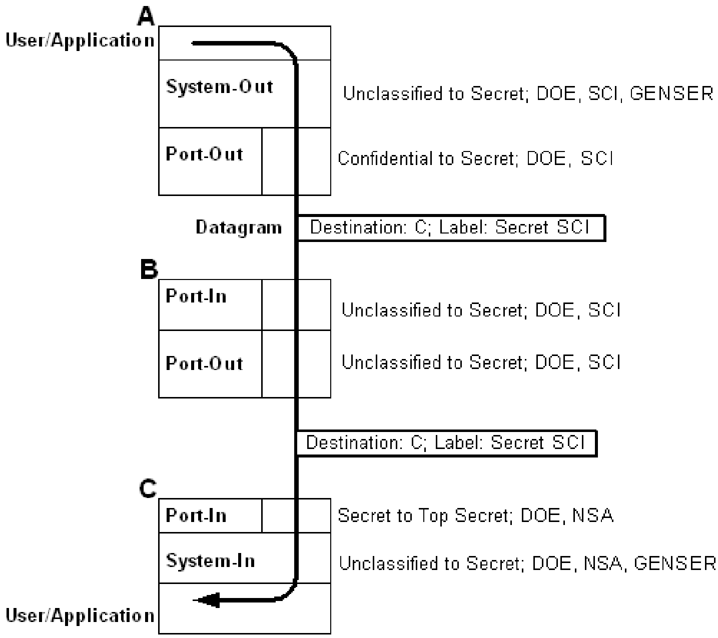
Example 23-3 in the previous section also shows what to do about unlabeled datagrams. The /RECEIVE_IMPLICIT_LABEL qualifier used on the second line ensures that the system accepts unlabeled received datagrams as if they were SECRET (see the `Implied... Receive:` output). If the received implicit label had been left at UNCLASSIFIED, unlabeled datagrams would not pass the incoming test.

Frequently, systems do not allow implicit labeling, to prevent unlabeled datagrams from entering (or passing through) a system.

Sample Implementation

Figure 23-1 shows a sample IPSO implementation.

Figure 23-1 Sample IPSO Implementation



Host A routes a datagram through host B to get to its destination, host C. The datagram has the label "SECRET SCI." This label passes A's outgoing system and port checks, as indicated. Since B only routes the datagram, it performs only port checks on the datagram. B then forwards the datagram to C. The datagram passes C's incoming port and system checks and successfully reaches the user or application.

The implementation assumes a cooperative network scenario. In this scenario, system managers on hosts A, B, and C communicate with each other on setting security options. Even intermediate hosts (like B) need to be aware of the nature of the secure traffic routed through. Otherwise, such a system could lose datagrams.

Other configurations may not set up system checks, and, therefore, not perform them. If an IPSO check fails, the datagram is out of range and the system sends an ICMP error message back to the originator. However, the ICMP message itself must have the appropriate label so that it can pass the system and port checks for an outgoing datagram.

Commands

You add IPSO labels to datagrams, and set system and line requirements for IPSO screening using commands in TCPware's Network Control Utility (NETCU). These commands are:

- SET IPSO
- SHOW IPSO

To invoke NETCU, enter:

```
$ NETCU
```

and enter the series of IPSO commands you need to configure the system or lines at the `NETCU>` prompts.

See Chapter 2, *NETCU Commands*, in the *NETCU Command Reference* for detailed descriptions of these commands.

You can also include each command in your TCPWARE:IPSO_SETUP.COM file.

The basic format of the NETCU command that sets IPSO screening is:

```
SET IPSO {/SYSTEM | /LINE} /LABEL-TYPE=(LEVEL=level, AUTHORITY=auth)
```

/SYSTEM or **/LINE=*line***	is the system-wide setting or the line number (such as `SVA-0`).
/LABEL-TYPE	is one of the following: **/LABEL**--label for both received (incoming) and transmitted (outgoing) datagrams **/IN_LABEL**--label for incoming datagrams only **/OUT_LABEL**--label for outgoing datagrams only
level	is either a single security level (***level***), or a range (minimum and maximum) security level in the form **(*min-level, max-level*)**. You can use either the name of the security level or its hexadecimal value as shown in Table 23-1; for example, `Top_Secret` or `%X3D`. In specifying the range of levels, the *min-level* must be lower on the list in Table 23-1 than the *max-level*.

auth	is either a single protection authority or a list of authorities in the form **(*auth1*, *auth2*, ..., *auth*n)**. You can use either the name of the authority or its hexadecimal value as shown in Table 23-2; for example, `GENSER` or `0x80`. Decimal and octal values are also acceptable. Each ***auth*n** can also be in the form **"*auth1*+*auth2*+...+*auth*n"** (a combination of authorities strung together with plus signs and enclosed in quotes). An example is `"GENSER+SCI"` – this means that you can process only datagrams with the *combination* GENSER+SCI. Alternatively, you can use the site-specific name set up in the IPSO_AUTHORITIES. file; for example, `ALPHA` for `"SCI+NSA"`. (See *Site-Specific Authority Names*.)

Suppose you want to set up line SVA-0 to accept all incoming or outgoing datagrams with a security level ranging from UNCLASSIFIED to SECRET. You also want the line to accept a protection authority of either GENSER or a combination of SCI+NSA. Perform the following command at the NETCU prompt:

```
NETCU> SET IPSO /LINE=SVA-0 -
_NETCU> /LABEL=(LEVEL=(UNCLASSIFIED,SECRET), -
_NETCU> AUTHORITY=(GENSER,"SCI+NSA"))
```

Adding an IPSO Label

Add explicit IPSO labels to transmitted datagrams using the /TRANSMIT_IMPLICIT_LABEL qualifier together with the /ADD qualifier, as in:

```
NETCU> SET IPSO /LINE=SVA-0 -
_NETCU> /TRANSMIT_IMPLICIT_LABEL=(LEVEL=SECRET, AUTHORITY=DOE, ADD)
NETCU> SHOW IPSO
%TCPWARE_NETCU-I-NOIPSODATA, no IPSO being processed on System
TCPware(R) for OpenVMS IPSO Configuration for line SVA-0:
Label       Level                        Authorities
-----       -----                        -----------
In:         UNCLASSIFIED to UNCLASSIFIED  None
Out:        SECRET to SECRET             DOE
Implied
  Receive:  UNCLASSIFIED                 None
  Transmit: SECRET                       DOE
ICMP Error: SECRET                       DOE
Adding implied label to transmitted datagrams
```

This imposes the SECRET level and DOE authority on all unlabeled datagrams going out on line SVA-0. The ADD keyword actually adds the IPSO header information to the datagrams.

Note from the SHOW IPSO output that the outgoing label and ICMP error requirements must already have been set to match the explicitly added label:

```
NETCU> SET IPSO /LINE=SVA-0 /OUT=(LEVEL=SECRET,AUTH=DOE)
NETCU> SET IPSO /LINE=SVA-0 /ERROR=(LEVEL=SECRET,AUTH=DOE)
Otherwise, the following error messages would appear:
%TCPWARE_NETCU-W-OUTOFRANGE, implied outgoing authority out-of-range for
SVA-0
%TCPWARE_NETCU-W-OUTOFRANGE, implied outgoing level out-of-range for SVA-0
%TCPWARE_NETCU-W-OUTOFRANGE, error authority out-of-range for SVA-0
%TCPWARE_NETCU-W-OUTOFRANGE, error level out-of-range for SVA-0
```

Accepting Datagrams Regardless of Authority

You can have the system accept datagrams regardless of their authority label by using the AUTHORITY=ANY value. In the following example, the host processes any datagram with a SECRET level and ignores the authority setting (note that the implicit labels and ICMP error were already set to match the line requirements):

```
NETCU> SET IPSO /LINE=SVA-0 /LABEL=(LEVEL=SECRET, AUTHORITY=ANY)
NETCU> SHOW IPSO
%TCPWARE_NETCU-I-NOIPSODATA, no IPSO being processed on System
TCPware(R) for OpenVMS IPSO Configuration for line SVA-0:
Label       Level               Authorities
-----       -----               -----------
In:         SECRET to SECRET    None
Out:        SECRET to SECRET    None

Implied
  Receive:  SECRET              None
  Transmit: SECRET              None
ICMP Error: SECRET              None
Ignoring authority field on received datagrams
Ignoring authority field on transmitted datagrams
```

Applying Implicit Labels

You can have the system associate labels with incoming or outgoing datagrams not having labels. Use the /RECEIVE_IMPLICIT_LABEL or /TRANSMIT_IMPLICIT_LABEL qualifier (without the ADD keyword), as in:

```
NETCU> SET IPSO /LINE=SVA-0 -
_NETCU> /RECEIVE_IMPLICIT_LABEL=(LEVEL=SECRET, AUTHORITY=GENSER)
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming authority out-of-range for
SVA-0
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming level out-of-range for SVA-0
NETCU> SHOW IPSO
%TCPWARE_NETCU-I-NOIPSODATA, no IPSO being processed on System
TCPware(R) for OpenVMS IPSO Configuration for line SVA-0:
```

```
Label        Level                          Authorities
-----        -----                          -----------
In:          UNCLASSIFIED to UNCLASSIFIED   None
Out:         UNCLASSIFIED to UNCLASSIFIED   None

Implied
  Receive:   SECRET                         GENSER
  Transmit:  UNCLASSIFIED                   None
ICMP Error:  UNCLASSIFIED                   None

%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming authority out-of-range for
SVA-0
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming level out-of-range for SVA-0
```

In the example, the system manager associated the "SECRET GENSER" option with any unlabeled incoming datagrams on line SVA-0. The system does not actually add the label to the datagram header. (The example also shows the error messages resulting from not setting up matching requirements for the incoming and outgoing labels.)

Additional keywords can accomplish the following:

REQUIRED	Require an IPSO label and do not use an implicit one
NOREQUIRED	Do not require an IPSO label and use an implicit one
NONE	Set an implicit label with an UNCLASSIFIED level and null authority

Datagrams with Extended Security Options

You can have the system accept or transmit datagrams containing Extended Security Options. TCPware enables this by default. To explicitly enable extended options, use the /EXTENDED_ALLOWED qualifier for incoming or outgoing datagrams, or both, as in:

```
NETCU> SET IPSO /LINE=SVA-0 /EXTENDED_ALLOWED
```

You can selectively disallow security options using the NOIN and NOOUT keywords (as in /EXTENDED_ALLOWED=NOIN), or disallow them more generally using /NOEXTENDED_ALLOWED. The latter produces the following lines at the bottom of a SHOW IPSO output:

```
Extended IPSO options on received datagrams disallowed
Extended IPSO options on transmitted datagrams disallowed
```

Stripping Datagrams of Options

You can have the system strip any Basic Security Options present in an outgoing datagram. This is useful for routers and forwarding datagrams on which you do not want to impose security restrictions. Use the /STRIP qualifier, as in:

```
NETCU> SET IPSO /LINE=SVA-0 /STRIP
```

This produces SHOW IPSO output that includes the line `Stripping IPSO option on transmitted datagrams`. /NOSTRIP is the default.

Be careful stripping datagrams of basic options. The datagrams could get lost as a result of further checks that require these options.

Setting IP Security Options First in the Datagram Header

Some security systems require that the security options appear first in datagram headers. If this is the case, use the /FIRST qualifier, as in:

```
NETCU> SET IPSO /LINE=SVA-0 /FIRST
```

This produces SHOW IPSO output that includes the line `Placing IPSO option first on transmitted datagrams`. /NOFIRST is the default.

Enabling ICMP Errors

If a datagram arrives out of range, the originator should receive a returned ICMP message to indicate the error. For this to happen, the returned message itself must have the proper security label. Enable this using the /ERROR_LABEL qualifier, as in:

```
NETCU> SET IPSO /LINE=SVA-0 -
_NETCU> /ERROR_LABEL=(LEVEL=SECRET, AUTHORITY=GENSER)
```

Define only a single level and authority for an error label. The level and authority should always be a subset of the level and authority defined for the outgoing label. For example, the following SHOW IPSO output shows a valid port setting for the requirement above:

```
TCPware(R) for OpenVMS IPSO Configuration for line SVA-0:

Label       Level                          Authorities
-----       -----                          -----------
In:         UNCLASSIFIED to UNCLASSIFIED   None
Out:        UNCLASSIFIED to SECRET         GENSER

Implied
  Receive:  UNCLASSIFIED                   None
  Transmit: SECRET                         GENSER
ICMP Error: SECRET                         GENSER
```

To disallow ICMP message handling, use /ERROR_LABEL=NONE. This resets the ICMP error setting to UNCLASSIFIED NONE. However, disallowing ICMP message handling can be risky.

Note! Always set the system requirements first. This sets each line identically. Then make isolated changes to the line requirements.

Automatic Startup

The STARTNET procedure uses the TCPWARE:ROUTING.COM file to automatically set security options at TCPware startup.

The file can contain the NETCU commands used to set IPSO options, as discussed earlier. Any commands in this file can refer to the IPSO_AUTHORITIES. file for authority definitions. (See *Site-Specific Authority Names*.)

Example 23-4 shows sample content of the ROUTING.COM file.

Example 23-4   Sample ROUTING.COM File

```
$ NETCU
SET IPSO /SYSTEM /LABEL=(LEVEL=(UNCLASSIFIED,SECRET),AUTH=(DOE,SCI))
SET IPSO /SYSTEM /RECEIVE_IMPLICIT=(LEVEL=UNCLASSIFIED,AUTH=DOE)
SET IPSO /SYSTEM /TRANSMIT_IMPLICIT=(LEVEL=SECRET,AUTH=DOE)
SET IPSO /SYSTEM /ERROR_LABEL=(LEVEL=SECRET,AUTH=DOE)
SET IPSO /LINE=SVA-0 /OUT_LABEL=(LEVEL=SECRET,AUTH=DOE)-
/IN_LABEL=(LEVEL=(UNCLASSIFIED,SECRET),AUTH=(DOE,SCI,"DOE+SCI"))
EXIT
```

The commands produce the following result:

```
NETCU> SHOW IPSO

TCPware(R) for OpenVMS IPSO Configuration for System:

Label       Level                     Authorities
-----       -----                     -----------
In:         UNCLASSIFIED to SECRET    DOE
                                      SCI
Out:        UNCLASSIFIED to SECRET    DOE
                                      SCI

Implied
  Receive:  UNCLASSIFIED              DOE
  Transmit: SECRET                    DOE
ICMP Error: SECRET                    DOE


TCPware(R) for OpenVMS IPSO Configuration for line SVA-0:

Label       Level                     Authorities
-----       -----                     -----------
In:         UNCLASSIFIED to SECRET    DOE
                                      SCI
                                      (DOE+SCI)
Out:        SECRET to SECRET          DOE
```

```
Implied
  Receive:  UNCLASSIFIED              DOE
  Transmit: SECRET                    DOE
ICMP Error: SECRET                    DOE
```

Site-Specific Authority Names

The `TCPWARE:IPSO_AUTHORITIES.` file (include the dot at the end) contains a table translating bit masks to protection authorities. Edit this file to create site-specific authorities ("aliases") or authority combinations you can associate with an alias name. You can create an alias authority name that is the name of the system a certain combination of authorities frequently protect. For example, you can associate the name ALPHA, the system that frequently needs to have SCI and NSA (inclusive) authority protection, with the authority combination "SCI+NSA".

To do this, enter the value ALPHA in the file along with the logically OR'd bit mask values of the combined pre-defined authorities, as indicated in Figure 43-2. In the case of ALPHA, the hexadecimal value `0x30` comes from `0x20` + `0x10`. In the case of DELTA, the hexadecimal value `0xC0` comes from `0x80` + `0x40` ("GENSER+SIOP-ESI").

Figure 23-2 Sample IPSO_AUTHORITIES. File

```
!GENSER    0x80
!SIOP-ESI  0x40
!SCI       0x20
!NSA       0x10
!DOE       0x08
!
!SITE-SPECIFIC:
ALPHA      0x30     !SCI+NSA
BETA       0x50     !SIOP-ESI+NSA
DELTA      0xC0     !GENSER+SIOP-ESI
```

As indicated in Figure 23-2, the predefined authorities (GENSER to DOE) are in the `IPSO_AUTHORITIES.` file for reference purposes only and are commented out using an exclamation point (`!`). You must also comment out the description after each site-specific entry.

CAUTION! Do not delete the `IPSO_AUTHORITIES.` file.

This setup allows you to specify a site-specific authority in an IPSO command such as:

```
NETCU> SET IPSO /LINE=SVA-0 /LABEL=(LEVEL=(UNCLASSIFIED,SECRET),-
_NETCU> AUTHORITY=(GENSER,ALPHA))
```

which, if combined with other appropriate settings, produces the SHOW IPSO output:

```
NETCU> SHOW IPSO
```

```
%TCPWARE_NETCU-I-NOIPSODATA, no IPSO being processed on System
TCPware(R) for OpenVMS IPSO Configuration for line SVA-0:
Label       Level                    Authorities
-----       -----                    -----------
In:         UNCLASSIFIED to SECRET   GENSER
                                     ALPHA(NSA+SCI)
Out:        UNCLASSIFIED to SECRET   GENSER
                                     ALPHA(NSA+SCI)

Implied
  Receive:  UNCLASSIFIED             GENSER
  Transmit: UNCLASSIFIED             GENSER
ICMP Error: UNCLASSIFIED             GENSER
```

Full SHOW IPSO Output

With the /FULL qualifier, the SHOW IPSO command produces the usual output along with additional counter information for incoming and outgoing datagrams. This information includes the number of datagrams that:

- Contain Basic Security Options
- Were delivered or transmitted
- Contain Extended Security Options
- Use implicit labeling
- Were rejected as out-of-range
- Were rejected due to containing extended options
- Lack a required basic option.

Typical output is:

```
NETCU> SHOW IPSO /FULL
TCPware(R) for OpenVMS IPSO Configuration for System:
Label       Level                         Authorities
-----       -----                         -----------
In:         UNCLASSIFIED to UNCLASSIFIED  None
Out:        UNCLASSIFIED to UNCLASSIFIED  None

Implied
  Receive:  UNCLASSIFIED                  None
  Transmit: UNCLASSIFIED                  None
ICMP Error: UNCLASSIFIED                  None

          Incoming datagrams screened by IPSO
              0 contained a BSO
            559 were delivered to receivers
              0 contained extended options
            559 used implicit labelling
```

```
          0 were rejected as out-of-range
          0 were rejected due to containing ESO
          0 lacked a required BSO

      Outgoing datagrams screened by IPSO
          0 contained a BSO
        100 were successfully transmitted
          0 contained extended options
          0 used implicit labelling
          0 were rejected as out-of-range
          0 were rejected due to containing ESO
          0 lacked a required BSO
```

Troubleshooting

Out-of-range datagram error messages appear both as warnings after SET IPSO and in SHOW IPSO output. Here are a few conditions that could produce out-of-range error messages:

- A label was set without an associated implicit label:

```
NETCU> SET IPSO /SYS /IN=(LEVEL=(CONFIDENTIAL,SECRET),AUTH=DOE)
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming authority out-of-range
for System
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming level out-of-range for
System
NETCU> SHOW IPSO
TCPware(R) for OpenVMS IPSO Configuration for System:
Label       Level                        Authorities
-----       -----                        -----------
In:         CONFIDENTIAL to SECRET       DOE
Out:        UNCLASSIFIED to UNCLASSIFIED None

Implied
  Receive:  UNCLASSIFIED                 None
  Transmit: UNCLASSIFIED                 None
ICMP Error: UNCLASSIFIED                 None

%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming authority out-of-range
for System
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming level out-of-range for
System
```

- An implicit label is not a subset of the default label setting:

```
NETCU> SET IPSO /SYS /REC=(LEVEL=CONFIDENTIAL,AUTH=DOE)
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming authority out-of-range
for System
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming level out-of-range for
System
NETCU> SHOW IPSO
TCPware(R) for OpenVMS IPSO Configuration for System:
```

```
Label       Level                         Authorities
-----       -----                         -----------
In:         UNCLASSIFIED to UNCLASSIFIED  None
Out:        UNCLASSIFIED to UNCLASSIFIED  None
Implied
  Receive:  CONFIDENTIAL                  DOE
  Transmit: UNCLASSIFIED                  None
ICMP Error: UNCLASSIFIED                  None

%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming authority out-of-range
for System
%TCPWARE_NETCU-W-OUTOFRANGE, implied incoming level out-of-range for
System
```

- An ICMP error label does not match the outgoing label setting:

```
NETCU> SET IPSO /SYS /ERROR=(LEVEL=CONFIDENTIAL,AUTH=DOE)
%TCPWARE_NETCU-W-OUTOFRANGE, error authority out-of-range for System
%TCPWARE_NETCU-W-OUTOFRANGE, error level out-of-range for System
NETCU> SHOW IPSO
TCPware(R) for OpenVMS IPSO Configuration for System:

Label       Level                         Authorities
-----       -----                         -----------
In:         UNCLASSIFIED to CONFIDENTIAL  SCI
Out:        SECRET to TOP_SECRET          SCI

Implied
  Receive:  CONFIDENTIAL                  SCI
  Transmit: SECRET                        SCI
ICMP Error: CONFIDENTIAL                  DOE

%TCPWARE_NETCU-W-OUTOFRANGE, error authority out-of-range for System
%TCPWARE_NETCU-W-OUTOFRANGE, error level out-of-range for System
```

PART VII Managing Additional Support

Chapter 24

PATHWORKS Support

Introduction

You can use TCPware for OpenVMS as a transport for Compaq Computer’s PATHWORKS product running between the OpenVMS system and a personal computer (PC).

You do not need to perform special configuration steps on either the OpenVMS system or the PC to use TCPware with PATHWORKS. However, the following information about PATHWORKS and TCPware may be helpful.

See *Troubleshooting* for troubleshooting tips.

PATHWORKS Version 4 Server

The VAX or Alpha system runs the PATHWORKS server software. It resides in the SYS$COMMON:[PCSA] directory. When you use TCPware and PATHWORKS, note the following:

- TCPware writes error logs to the SYS$SYSROOT:[PCSA] directory.
- Make sure that you start TCPware before starting PATHWORKS.
- Make sure that you use the following command when you start PATHWORKS on the VAX or Alpha system:

  ```
  $ @SYS$STARTUP:PCFS_STARTUP TCP
  ```

  After you start PATHWORKS on your VAX/Alpha system:

- The OpenVMS DCL command SHOW SYSTEM lists two processes:
  - NBNS
  - PCFS_SERVER
- The SHOW CONNECTIONS command in TCPware Network Control Utility (NETCU) shows the following two PATHWORKS-related services:

ID [1]	RecvQ	SendQ	Local address...	Foreign address...	State...
BG*nn*	0	0	*.netbios-ssn	*.*	LISTEN
BG*nn*	0	0	*.netbios-ns	*.*	
[1] The actual values may be different					

PATHWORKS Version 5 Server

The VAX or Alpha system runs the PATHWORKS server software. When you use TCPware and PATHWORKS, note the following:

- Make sure that you start TCPware before starting PATHWORKS
- Configure the PATHWORKS Server to use TCP/IP as a transport

After you start PATHWORKS on the VAX/Alpha:

- The OpenVMS DCL command SHOW SYSTEM lists the PATHWORKS master process PWRK$MASTER and several other PATHWORKS processes.
- The TCPware SHOW CONNECTIONS command in NETCU shows the following PATHWORKS-related services:

ID [1]	RecvQ	SendQ	Local address...	Foreign address...	State...
INET*nn*	0	0	*.netbios-ssn	*.*	LISTEN
INET*nn*	0	0	*.netbios-ns	*.*	
INET*nn*	0	0	*.netbios-dgm	*.*	
[1] The actual values may be different					

On the PC

TCPware and PATHWORKS were tested together with several third party PC TCP/IP products and LAN adapters, including the following:

- PATHWORKS for DOS (TCP/IP) software with a Compaq DEPCA adapter
- Ungermann-Bass's LAN Manager 2.0 and TCP BNS 16.51 software with an Ungermann-Bass NIUpc/EOTP[1] adapter

When you install and configure the PC software:

- PATHWORKS Version 4 updates the CONFIG.SYS and AUTOEXEC.BAT files.
- PATHWORKS Version 4 and Version 5 create either a PROTOCOL.INI or LANMAN.INI file.
- Some PC software products may require that you specify the VAX server name in all uppercase when you use the USE or NET USE command.
- When using Ungermann-Bass's PC software, you may need to use the NBUTIL utility to specify the name and internet address of the OpenVMS system.

[1] NIUpc is a registered trademark of Ungermann-Bass, Inc.

Troubleshooting

Version 4

If you cannot make a complete connection between the workstation and the OpenVMS system using PATHWORKS Version 4, do the following:

- Check the TCPWARE:HOSTS. file on the OpenVMS system. Make sure that the workstation's name and internet address are in the file.
- Do a SHOW SYSTEM on the OpenVMS system. Make sure that the NBNS and PCFS_SERVER processes are running; if not, start PATHWORKS on the OpenVMS system.
- Issue the NETCU SHOW CONNECTIONS command on the OpenVMS system. Verify that the `*.netbios-ssn` and `*.netbios-ns` services are listed.
- If these services are not listed, make sure that you start TCPWare before PATHWORKS.
- Look in the SYS$SYSROOT:[PCSA] directory on the OpenVMS system, to see if any of the error log files contain useful information.
- On the workstation, check that the PROTOCOL.INI or LANMAN.INI file has the correct configuration information for your workstation.
- On the workstation, verify that the syntax of your USE or NET USE command is correct. (Note the use of uppercase characters.)
- Make sure that the service specified in the USE or NET USE command is listed in the SYS$SYSROOT:[PCSA]PCFS$SERVICE_DATABASE.DAT file on the OpenVMS system.

Version 5

If you cannot make a complete connection between the workstation and the OpenVMS system using PATHWORKS Version 5, do the following:

- Make sure TCPware knows how to map the workstation's host name to its internet address. Check this using NSLOOKUP *hostname*. NSLOOKUP should report the internet address for the host. If it does not, update your primary DNS server so it can resolve the workstation name (or add an entry for the workstation in the OpenVMS system's TCPWARE:HOSTS. file).

- Do a SHOW SYSTEM on the OpenVMS system. Make sure that the PWRK$MASTER, NETBIOS, and other PWRK$ processes are running. If they are not, start PATHWORKS on the OpenVMS system.
- Issue the NETCU SHOW CONNECTIONS command on the OpenVMS system. Verify that the `*.netbios-ssn`, `*.netbios-ns`, and `*.netbios-dgm` services are listed. If these services are not listed, make sure that you start TCPware before PATHWORKS.
- Look in the PWRK$ROOT:[LOGS] directory on the OpenVMS system to see if any of the error log files contain useful information. See specifically the file PWRK$KNBDAEMON_*node*.LOG, where *node* is the particular PATHWORKS node. This contains the TCP/IP binding information.
- On the workstation, check that the PROTOCOL.INI or LANMAN.INI file has the correct configuration information for your workstation.
- On the workstation, verify that the syntax of your USE or NET USE command is correct. (Note the use of uppercase characters.)
- Make sure that the service specified in the USE or NET USE command is available on the OpenVMS system.

Chapter 25

Tunneling DECnet over IP

Introduction

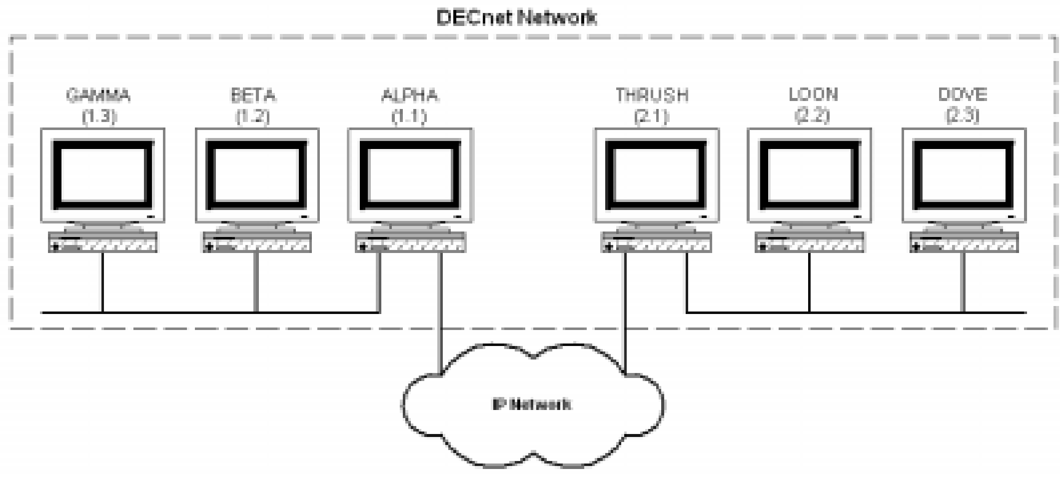
This chapter describes how to set up and manage the TCPware for OpenVMS feature that supports tunneling of DECnet over IP-based networks.

DECnet over IP support provides tunneling of DECnet data link layer packets over a TCP/IP connection between two OpenVMS systems. Both systems must run TCPware for OpenVMS. This feature lets you establish a DECnet line and circuit between two OpenVMS systems connected to each other over a TCP/IP network.

Note! This feature is intended for DECnet Phase IV. There is no need to use it with DECnet/OSI (DECnet Phase V). Instead, configure DECnet/OSI to run over TCP/IP as described in the DECnet/OSI documentation. No special TCPware configuration is required.

Figure 25-1 shows an example of one of the many ways that you can use tunneling DECnet over IP. The example shows two DECnet areas connected together through a DECnet over IP tunnel. The shaded nodes, ALPHA and THRUSH, run both DECnet and TCPware for OpenVMS. The other nodes may run only DECnet.

Figure 25-1   Tunneling DECnet over IP



From the system manager's point of view, DECnet over IP support consists of:

- CNFNET prompts that configure and start up DECnet over IP tunnels
- Options to STARTNET and SHUTNET that start and shut any DECnet over IP tunnels
- NETCU SHOW DNIP command to show information about any DECnet over IP tunnels
- Device driver DNIPDRIVER

DECnet over IP Lines

To use DNIP lines, you must install a DECnet routing license on the system, unless the DNIP line is the only DECnet line on the system.

DECnet over IP Tunnels

You invoke DECnet over IP configuration in one of two ways:

- During TCPware for OpenVMS configuration with CNFNET, by answering `YES` at the `Do you want to configure DECnet over IP tunnels?` prompt

  At any time after configuration, by invoking just the DECnet over IP portion of configuration by entering this command: **`@TCPWARE:CNFNET DNIP`**

CNFNET prompts for the following information for each DECnet over IP tunnel:

- DECnet line name (for example, DNIP-0-0)
- Remote host name or IP address
- Port number of the tunnel on the local host

- Port number of the tunnel on the remote host (this is optional)

The port number is the TCP port number at which the system establishes DECnet over IP connections. You can reuse this same port number for all tunnels on a single system, and for all systems in your network. Note that the default value (64215) has no special significance, it is simply a "random" TCP port that any other service on your system is unlikely to use.

See the the *Installation & Configuration Guide*, Chapter 4, *Configuring the TCP/IP Services*, the *Configure the DECnet over IP Tunnels* section.

When you start TCPware, it automatically:

- Starts up the configured DECnet over IP tunnels.
- Performs DECnet configuration of the associated DECnet lines and circuits.

You need to perform all other DECnet network management through NCP, for example: defining DECnet node numbers and names.

Starting and Stopping

Normally, all configured DECnet over IP tunnels start when you start TCPware for OpenVMS. If for any reason you want to shut down and later restart, only the DNIP tunnels (and not all of TCPware for OpenVMS), use the following commands:

- To stop all DECnet over IP tunnels configured on your host: **`@TCPWARE:SHUTNET DNIP`**
- To start all DECnet over IP tunnels configured on your host: **`@TCPWARE:STARTNET DNIP`**

Status

Use the SHOW DNIP command in the Network Control Utility (NETCU) to show information about the currently configured DECnet over IP tunnels.

See the SHOW DNIP command of the *NETCU Command Reference*.

Troubleshooting

If you get the following message when configuring the DECnet over IP tunnel, you may have entered a host domain name instead of an IP address. It is possible that your `HOSTS.` file or DNS could not resolve the domain name. Try re-specifying the host address as an IP address:

```
%DNIP-E-INVIA, invalid internet address
```

Chapter 26

X Display Manager Server

Introduction

TCPware's X Display Manager (XDM) Server manages remote X displays (X terminals). When starting up, remote X displays communicate with XDM through the UDP-based X Display Manager Control Protocol (XDMCP). The XDM Server creates a DECwindows login process that prompts remote X display users to log in and create a DECwindows session.

TCPware's XDM Server supports broadcast, direct, and indirect requests. It also supports display authentication. A direct request is to a particular display or displays, and a broadcast request polls any number of displays for a response. An indirect request forwards the request to another XDM server to determine the X display. Display authentication requires that the XDM Server and the X display share a private key.

The Server is supported on all versions of the Alpha platform and on VAX/VMS 6.1 or higher. VAX/VMS 5.5-2 is supported if DEC C or the DEC C Run-Time Library (RTL) is installed.

The XDM Server is based on the X11R6.1 release from X Consortium.

This chapter describes the XDM Server, its configuration, and how to impose access restrictions from other servers.

Installation and Initial Setup

Enable the XDM Server through CNFNET configuration by responding with YES at the following prompts (both default to NO):

- Do you want to use the TCPware for OpenVMS XDM Server:
- Do you want to restart XDM:

When enabled and started, the XDM Server runs as a detached process, TCPware_XDM. A log file, TCPWARE:XDMSERVER.LOG, is created that captures log information enabled through the `DisplayManager.debugLevel` X Resource (see the next section).

Server Configuration

Configure the XDM Server using an X Resource style configuration file, TCPWARE:XDM_CONFIG.DAT. You must create this file. (A template is provided as TCPWARE:XDM_CONFIG.TEMPLATE, which you can rename TCPWARE:XDM_CONFIG.DAT.)

Example 26-1 shows a sample configuration file.

Example 26-1   Sample XDM_CONFIG.DAT File

```
# TCPWARE:XDM_CONFIG.DAT
#
DisplayManager.debugLevel:        1
DisplayManager.accessFile:        TCPWARE:XDM_ACCESS.DAT
DisplayManager.removeDomainname:  false
DisplayManager.keyFile:           TCPWARE:XDM_KEYS.DAT
#
```

Any characters after the pound sign (`#`) are ignored. The XDM Server supports the following X Resources:

- **`DisplayManager.debugLevel`**
- **`DisplayManager.opcomLevel`**

  If these resource values are set to non-zero, this enables logging. The default is **`0`** (no logging). If set to non-zero, the XDM Server logs debugging information to the TCPWARE:XDMSERVER.LOG file or sends a message to OPCOM. The debug levels are as follows:

0	None	2	Critical	4	Warning	6	Informational
1	Alert	3	Error	5	Notice	7	Debug

  The defaults are:

  – **`DisplayManager.debugLevel:4`**
  – **`DisplayManager.opcomLevel:5`**

- **`DisplayManager.accessFile`**

  Specifies the access control file that defines access limitations of servers to the XDM Server. By default no file is specified, which allows any X display to connect to the XDM Server. If this resource is specified with a non-existent or empty file, no X display can connect to the XDM Server. The usual file is TCPWARE:XDM_ACCESS.DAT. See *Server Access Control*.

- **`DisplayManager.removeDomainname`**

  When computing the display name for XDMCP clients, the name resolver typically creates a fully qualified hostname for the terminal. As this may be confusing, you can use this resource to remove the domain name portion of the hostname, if it is the same as the domain name of the local host when this variable is set. The value is **`true`** by default.

- **`DisplayManager.keyFile`**

  XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between XDM and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key.

  The display name is either a unique name associated with the display hardware (such as `-Ethernet-8:0:2b:a:f:d2` in the form *manufacturer-model-serial*) or any unique identifier. The shared key is a 56-bit integer represented by a 14-digit hexadecimal integer prefixed with **`0x`**.

  Example 26-2 shows a sample key file. (A template file, XDM_KEYS.TEMPLATE, is provided that you can rename XDM_KEYS.DAT and use as the default.)

Example 26-2 Sample XDM_KEYS.DAT File

```
# TCPWARE:XDM_KEYS.DAT
#
# display name            shared key (14 digit hex)
# ----------------------  ------------------------
  -Ethernet-8:0:2b:a:f:d2  0x4f098af322dd98
#
```

Server Access Control

You can control access to the XDM Server in the TCPWARE:XDM_ACCESS.DAT file by enabling the **`DisplayManager.accessFile`** resource in the XDM_CONFIG.DAT file (see *Server Configuration*). (A template file, XDM_ACCESS.TEMPLATE, is provided that you can rename XDM_ACCESS.DAT and use as the default.)

Note that if you specify the `DisplayManager.accessFile` resource without a filename value, any X display can connect to the XDM Server. If the file specified is empty or nonexistent, no X display can connect to the XDM Server.

The XDM_ACCESS.DAT file contains entries that control the response to direct and broadcast queries, and contains separate entries for indirect queries. Each entry is a hostname, pattern, or macro.

- For a hostname, all comparisons involve network addresses, so that you can use any name that converts to the correct network address. An exclamation point (**!**) preceding a hostname excludes that host.
- A pattern involves wildcards, where only canonical hostnames are compared – do not attempt to match aliases. An exclamation point (**!**) preceding a pattern excludes hosts that

match the pattern. A pattern includes one or more of the following wildcard meta-characters:

*****	Matches any sequence of characters compared against the hostname
?	Matches any single character compared against the hostname

- A macro definition contains a macro name preceded by **%** and a list of hostnames or other macros, which can be nested.

An indirect entry also contains a hostname or pattern, but follows it with a list of hostnames (such as defined by a macro) to which indirect queries should be sent. When checking access for a particular display host, each entry is scanned in turn and the first matching entry determines the response. Direct and broadcast entries are ignored when scanning for an indirect entry, and indirect entries are ignored when scanning for direct or broadcast entries.

Other access control file formats are as follows:

Blank lines are ignored	
#	Comment delimiter, causing the rest of that line to be ignored
****	Line continuation character, allowing indirect host lists to span multiple lines

Example 26-3 shows a sample XDM_ACCESS.DAT file. Note that the positioning of entries is important. For example, if the `!bogus.bart.com` and `*.bart.com` lines had been transposed, direct or broadcast access from `bogus` would have been allowed, which is not desired. In the entry `!bogus.bart.com dummy`, the `dummy` is a dummy host.

Example 26-3   Sample XDM_ACCESS.DAT File

```
# TCPWARE:XDM_ACCESS.DAT
#
# Direct/broadcast query entries
#
bambi.marge.com    # allow access from bambi on marge
!bogus.bart.com    # disallow direct/broadcast service from bogus
*.bart.com         # allow access from any display in bart (x bogus)
#
# Indirect query entries
#
%HOSTS  expo.bart.com  xenon.bart.com  excess.bart.com HR>
kanga.bart.com
extract.bart.com  xenon.bart.com  # force extract to contact xenon
!bogus.bart.com   dummy           # disallow indirect access to bogus
*.bart.com        %HOSTS          # all others get to choose from
#                                   hosts in the %HOSTS macro list
#
```

Chapter 27

DECwindows Transport Interface

Introduction

This chapter describes how to configure and manage the TCPware DECwindows Transport Interface if you are running OpenVMS Version 5.5-2.

You may want to run local DECwindows applications remotely, or remote X Window System applications locally. If you are running OpenVMS Version 5.5-2 (the minimum supported version of OpenVMS), you need to configure the DECwindows Transport Interface. If you have a later version of OpenVMS, simply use the transport interface supplied with Compaq's DECwindows product.

Note! The TCPware DECwindows Transport Interface is not available on OpenVMS Alpha V1.5 and OpenVMS VAX V6.1 (and later) systems. Use Compaq's DECwindows Transport Interface provided with DECwindows instead.

Setting Up the Interface

To configure the DECwindows Transport Interface, the remote system must have X Window System and TCP/IP support.

Check the SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM file that it contains the proper information. The transport name in the file can be either **`TCPWARE`** or **`TCPIP`**. If you are running OpenVMS V5.5-2, you can choose between using the TCPDRIVER or BGDRIVER transport interface:

- If you enter **`TCPWARE`**, the server uses the TCPware transport (TCPDRIVER)
- If you enter **`TCPIP`**, the server uses the UCX Compatibility Services (BGDRIVER)

If you are running OpenVMS Alpha V1.5 or OpenVMS VAX V6.1 and later, make sure to configure **`TCPIP`** as the DECwindows transport name instead of **`TCPWARE`**. The TCPware interface is not available with the latter versions of OpenVMS.

To use the TCPware DECwindows Transport Interface, perform these steps:

1	Configure the DECwindows Transport Interface during TCPware configuration, or at a later date, by entering the following command: `$ @TCPWARE:CNFNET DECW`

2	If the SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM file does not exist, create it by copying the DECW$PRIVATE_SERVER_SETUP.TEMPLATE file to *.COM. Then edit the file as follows:
	a Locate the following commented-out line in the file:
	`$ ! DECW$SERVER_TRANSPORTS ==`
	(Ignore the `DECW$SERVER_TRANSPORTS` line in the later `$DO_TCPIP` section.)
	b Remove the comment character (!) and:
	• If configuring the TCPware transport, add the following to the line:
	`$ DECW$SERVER_TRANSPORTS == "DECNET,LOCAL,LAT,TCPWARE"`
	• If configuring for UCX Compatibility Services, add the following to the line:
	`$ DECW$SERVER_TRANSPORTS == "DECNET,LOCAL,LAT,TCPIP"`

3	If you are running DECnet, be sure to start the products in the order DECnet, TCPware, and DECwindows. (In any case, start TCPware before starting DECwindows.)
4	Restart DECwindows: `$ @SYS$MANAGER:DECW$STARTUP RESTART` If the DECwindows Transport Interface does not work, see *Troubleshooting*.

Setting Up the Remote Host

For security reasons, you usually must configure the target display host to allow incoming X Window System applications from the OpenVMS system host. You may need to enter the OpenVMS host name in a file on the target display host. Check the remote host system’s documentation for details.

If the remote host is another OpenVMS system, its SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM file should have the same changes as described in the previous section.

If the remote host is not an OpenVMS system, configure "security" there to allow incoming connections on the currently active session. Check the remote host system’s documentation for details.

Displaying on a Remote Host

To display a locally run DECwindows application on a remote host:

1	Verify that your system manager configured the target display host to accept incoming X Window System connections.

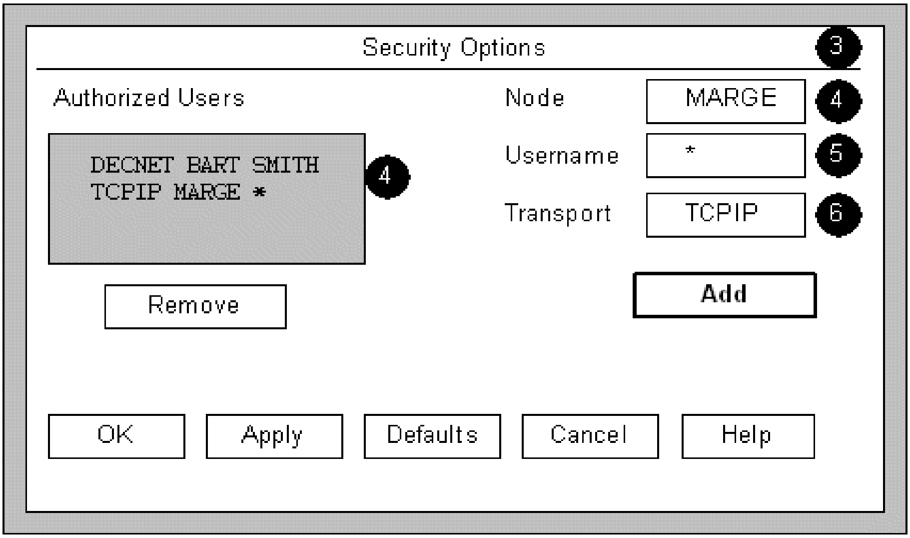
2	When you are ready to run your DECwindows application, enter the following commands at the DCL prompt:
	`$ SET DISPLAY/CREATE/NODE=remote-node/TRANSPORT=TCPIP`
	`$ RUN local-application`
	where *remote-node* is the remote workstation on which you want the application to display, and *local-application* is the name of the local application you want to display on the remote host.
	For example, to display the DECW$CALC application on host DAISY:
	`$ SET DISPLAY/CREATE/NODE=DAISY/TRANSPORT=TCPIP`
	`$ RUN SYS$SYSTEM:DECW$CALC`

Displaying Locally

If you want to run a DECwindows application on a remote host and display it locally on the OpenVMS system running TCPware (see the matching items in the example shown in Figure 27-1):

1	Verify that your system manager configured DECW$PRIVATE_SERVER_SETUP.COM to use the TCPware transport or the TCP/IP transport. You need to know which transport is being used for step 6 of this procedure.
2	Verify that the Session Manager is running on the local system.
3	Select the `Security...` option from the Session Manager's `Options` menu.
4	Enter the remote node name in the security database next to the `Node` field. If you want to display the application on your own node, use a zero (**`0`**) for the node name. If the remote node is a new entry, add it first so that it appears in the `Authorized Users` field, in the format *transport node username*.
5	Enter an asterisk (**`*`**) as the `Username` (you cannot restrict sessions based on usernames with TCP/IP).
6	Enter **`TCPIP`** or **`TCPware`** as the `Transport` based on how you set the transport in the SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM file, as described in the *Setting up the DECwindows Transport Interface* section.

Figure 27-1   DECwindows Security Options Screen



Note! The security records in the database are OpenVMS account-specific. If you enter the remote node name and **TCPIP** while you are in OpenVMS account SMITH, someone must be logged in as SMITH at the time you try to display the remote application.

See your DECwindows or Motif documentation for more information. Also see the remote host documentation for details on how to direct the application's display to the OpenVMS system.

Troubleshooting

If the DECwindows Transport Interface is not working:

- Reinstall TCPware if you originally installed TCPware on VMS or OpenVMS Versions 5.0, 5.1 or 5.2, and upgraded to Version 5.3 or later. You need only reinstall TCP-OpenVMS, and not the FTP, TELNET, SMTP, or NFS products. Then reconfigure the DECwindows Transport Interface.

  After reinstalling TCP-OpenVMS, you will have these files on your OpenVMS system:

  ```
  SYS$SHARE:[SYSLIB]DECW_TRANSPORT_TCPWARE.EXE
  SYS$COMMON:[TCPWARE]DECW_CONTROL.COM
  ```
- Make sure you have basic connectivity between the two systems in both directions. You can use any utility, such as TELNET or PING, that is supported on both systems.
- Check the `TCPWARE:HOSTS.` file on the OpenVMS system and make sure the other system's internet address is in it.
- Make sure "Security" is configured to allow incoming connections on the currently active

session. (See your DECwindows or Motif documentation for information.)

- Use the NETCU SHOW CONNECTIONS command on the TCPware for OpenVMS system to make sure the server is listening for incoming connections.
- You should see an entry for `*.6000` in the `Local Address` column, in the `LISTEN` state. If this entry is not there, double-check the configuration steps above.
- Check if the SYS$MANAGER:DECW$SERVER_*x*_ERROR.LOG file exists, and see if it contains any useful messages.

  Error messages about fonts or font files usually indicate that the target system does not have the necessary fonts to display the application.
- Double-check your OpenVMS system startup file to be sure that DECwindows starts *after* TCPware.

PART VIII Network Testing Tools

Chapter 28

Network Testing Tools

Introduction

This chapter describes the following network testing tools included with TCPware:

CHARGEND	Sends strings of characters, ignoring data received
DAYTIMED	Sends the current date and time
DISCARD	Discards any data it receives
ECHOD	Returns any data it receives
FINGER	Extracts user information from a remote user information program
IDENT	Other servers use this to determine the user associated with a connection
NETCU DEBUG	TCPware's NETCU provides debugging commands
NSLOOKUP	Verifies DNS name server and host domain information
PING	Verifies that a host on the network is up and reachable
QUOTED	Checks for TCP connectivity between hosts
TCPDUMP	Tracks TCP packets over the network
TIME	Time service
TRACEROUTE	Tracks TCP routes over the network

CHARGEND

CHARGEND is the Character Generator Protocol server, defined in RFC 864. It sends strings of characters, ignoring data received, and is a useful debugging tool. CHARGEND can be TCP- or UDP-based. STARTNET starts the service automatically using the TCPWARE:MISC_CONTROL.COM file. The TCPWARE_COMMON:

[TCPWARE.EXAMPLES] directory provides the source code. You can test this service through a TELNET connection to it.

DAYTIMED

DAYTIMED is the Daytime Protocol server, defined in RFC 867. It sends the current date and time as a character string without regard to input. It is a useful debugging and measurement tool. DAYTIMED can be TCP- or UDP-based. STARTNET starts the service automatically using the TCPWARE:MISC_CONTROL.COM file. The TCPWARE_COMMON:[TCPWARE.EXAMPLES] directory provides the source code. You can test this service through a TELNET connection to it.

DISCARD and DISCARDD

DISCARD is the Discard Protocol client and DISCARDD is the Discard Protocol server, defined in RFC 863. It simply throws away any data it receives, and is a useful debugging and measurement tool. It can be TCP- or UDP-based. STARTNET starts the service automatically using the TCPWARE:MISC_CONTROL.COM file. The TCPWARE_COMMON:[TCPWARE.EXAMPLES] directory provides the source code for DISCARD and DISCARDD. You can test the service using a TELNET connection to it.

ECHOD

ECHOD is the Echo Protocol server, defined in RFC 862. It returns any data it receives and is a useful debugging and measurement tool. It can be TCP- or UDP-based. STARTNET starts the service automatically using the TCPWARE:MISC_CONTROL.COM file. You can test this service using a TELNET connection to it.

FINGER and FINGERD

FINGER is the Finger User Information Protocol, defined in RFC 1288. It is used to extract user information from a remote user information program (RUIP). FINGER is TCP-based. STARTNET starts the service automatically using the TCPWARE:MISC_CONTROL.COM file. The TCPWARE_COMMON:[TCPWARE.EXAMPLES] directory provides the source code for FINGERD.

Use FINGER at the DCL level as follows:

```
$ FINGER user@host
```

The *user* is the user to extract information about on the specified *host*. You must use both parameters.

FINGERD reports the day of the week, hours, and minutes of a login time only if the user has been logged in for LESS than seven days. If the user has been logged on for MORE than seven days, the date in the format dd-mm-yy is reported.

Note! Although a two-digit year is reported, it is compliant with the year-two-thousand reporting. For example:

Old format:
SMITH ZEUS Mary Smith 0 Fri 14:36 FTA419:

New format:
JONES ZEUS John Jones 14 10-Dec-97 FTA372:

IDENT

Other servers use the Identification server (formerly the Authentication Server) to determine the user associated with a connection. The MISC_CONTROL command procedure automatically starts the IDENT server.

NETCU DEBUG

TCPware's Network Control Utility (NETCU) includes DEBUG commands for the IP, TCP, and UDP layers. They display information about IP datagrams, TCP segments, and UDP datagrams sent and received over the network. You can use them to debug network problems.

See the *NETCU Command Reference* for the DEBUG IP, DEBUG TCP, and DEBUG UDP command descriptions.

NSLOOKUP

The NSLOOKUP utility sends test queries to a DNS name server to test the DNS configuration. You would use NSLOOKUP to find information such as the following for a selected host or domain:

- Default name server
- DNS database records, such as all, A, CNAME, and MX
- All the hosts in a domain
- User processes

The NSLOOKUP utility has two modes, noninteractive and interactive:

Noninteractive mode	Query a host and exit NSLOOKUP
Interactive mode	Query a host and enter further NSLOOKUP commands

Setting the mode depends on the syntax you use in entering the NSLOOKUP command. You access noninteractive mode if you use the following syntax at the DCL prompt:

```
$ nslookup host-to-find [server-to-query]
$
```

You access interactive mode if you use the following syntax at the DCL prompt:

```
$ nslookup [- server-to-query]
>
```

The difference is that noninteractive mode requires at least the *host-to-find* parameter, while interactive mode is signalled by either the lack of an parameter altogether or a dash preceding the *server-to-query* parameter. Also, after a noninteractive query, you are back at the DCL prompt, while the interactive query puts you at the > prompt so that you can enter further NSLOOKUP commands.

Both modes allow you to enter options, as described in *Setting Options*.

Noninteractive Mode

The full noninteractive mode syntax is as follows:

```
$ nslookup [-option [-option...]] host-to-find [server-to-query]
```

-option	Sets the state information for a group of lookups. (See *Setting Options* for details on state information.) You can have multiple options, each prefixed by a dash and separated by a space.
host-to-find	Hostname or IP address of the host about which to seek information.
server-to-query	Hostname or IP address of the DNS server to query. You can specify a server only if you also specify a *host-to-find*. If omitted, TCPware uses the current default server.

Enter the **nslookup** command and options in lowercase, but enter the hostname and server name values in their proper case. To interrupt noninteractive mode, use **Ctrl/C**.

Interactive Mode

The full interactive mode syntax is as follows:

```
$ nslookup [-option [-option...]] [- server-to-query]
```

```
> { host-to-find | command }
```

-option	Sets the state information for a group of lookups. (See *Setting Options* for details on state information.) You can have multiple options, each prefixed by a dash and separated by a space.
- server-to-query	Hostname or IP address of the DNS server to query. Precede the parameter with a dash and a space.
host-to-find	Hostname or IP address of the host about which to seek information, as entered at the NSLOOKUP prompt (>).
command	One of the query commands listed in the *Query Command Reference* section. The command must be less than 256 characters; its case does not matter.

Enter the **nslookup** command and options in lowercase, but enter the hostname or server name in its proper case. The **nslookup** command alone always returns the default server information before presenting the > prompt, at which you can enter either the *host-to-find* or a *command*. To interrupt interactive mode, use **Ctrl/C**. To exit, enter **exit** or **Ctrl/Z**.

Entering Host-to-Find Names

You can enter the *host-to-find* as a hostname or an IP address. If you enter an IP address, the hostname is returned by default.

To look up a host not in the current domain, add a trailing period, such as:

```
$ nslookup iris.flower.com.
```

If the trailing period is missing, the default domain name (as determined by the **set domain**, **set srchlist**, **set defname**, or **set search** commands) is appended.

See the **set** command description in the *Query Command Reference*.

Setting Options

NSLOOKUP options set the state information for the particular query. You can set options using the -*option* parameter, or in interactive mode using the **set** command. Each option corresponds to a **set** command keyword.

See the **set** command description for the keywords that correspond to the options.

For example, the following command uses -*option* parameters to set the `querytype` and `timeout` parameters for the next series of interactive lookups to server AMOS.

```
$ nslookup -querytype=hinfo -timeout=10 - amos
```

The alternative would be:

```
$ nslookup
```

```
> set querytype=hinfo
> set timeout=10
> amos
```

The following command tries to get information about node SIRIUS in domain NENE.COM by querying server ANDY. If unsuccessful, NSLOOKUP tries ten more times and returns to the DCL prompt.

```
$ nslookup -domain=nene.com -retry=10 sirius andy
```

Query Command Reference

This section describes the NSLOOKUP query commands accessible at the **>** prompt. Enter the commands in lowercase or they are considered to be uppercase *host-to-find* entries.

exit

Exits NSLOOKUP.

Format

> exit

Enter the command in lowercase or it is considered to be hostname "EXIT".

Synonym

> **Ctrl/Z**

Example

This example displays information about the server and exits NSLOOKUP.

```
$ nslookup
Default Server:  sirius.nene.com
Address:  192.168.95.1
> exit
$
```

finger

Connects with the FINGER server on the (previously defined) host to display FINGER information. With the optional *username* parameter, you can focus the lookup on one user.

You can redirect the output from the **`finger`** command to a file using the greater-than symbol, as follows:

```
finger > filename.ext
finger >> filename.ext
```

The **>** places the output into *filename.ext*. The **>>** appends the output onto file *filename.ext*. When you direct output to a file, NSLOOKUP prints hash marks after every 50 records received from the server.

Format

> **finger *[username]***

Enter the command in lowercase or it is considered to be hostname "FINGER".

Parameter

username

Focuses the lookup to one username. The *username* can be case-sensitive.

Example

This example defines the current host as SIRIUS and connects with the FINGER server on SIRIUS to display FINGER information on user GANDALF.

```
$ nslookup
> sirius
Default Server:  sirius.nene.com
Address:  192.168.95.1

Name:    sirius.nene.com
Address:  192.168.95.1

> finger gandalf
[sirius.nene.com]
Status of VMScluster on node NENE at 23-JUN-1999 16:52:04.62
Username   Node     Name         New Mail   When        Terminal
GANDALF    SIRIUS   Gandalf                 3 Mon 08:34 FTA1211:
GANDALF    SIRIUS   Gandalf                 3 Mon 08:34 FTA1212:
```

help

Displays a brief summary of NSLOOKUP commands.

Format

> **help**

Enter the command in lowercase or it is considered to be hostname "HELP".

Synonym

> **?** (question mark)

Example

> **?**

```
                      NSLOOKUP COMMAND SUMMARY
      (identifiers are shown in uppercase, [] means optional)
Command          Meaning
---------------------------------------------------------------------
NAME          - print info about the host/domain NAME using default server
NAME1 NAME2   - as above, but use NAME2 as server
ls [opt] DOMAIN [> FILE] - list entities in DOMAIN (optional: output to
FILE)
  -a          - list canonical names and aliases
  -h          - list HINFO (CPU type and operating system)
  -s          - list well-known services
  -d          - list all records
  -t TYPE     - list records of the given type (e.g., A,CNAME,MX, etc.)
view FILE     - sort an 'ls' output file and view it
help or ?     - display this screen
spawn or ! [command] - create child process and execute command
finger [USER] - finger the optional NAME at the current default host
root          - set current default server to the root
server NAME   - set default server to NAME, using current default server
lserver NAME  - set default server to NAME, using initial server
set OPTION     - set an option

  all         - print options, current server and host
  [no]defname - append domain name to each query
  [no]recurse - ask for recursive answer to query
  [no]vc      - always use a virtual circuit
  domain=NAME - set default domain name to NAME
  srchlist=N1[/N2/.../N6] - set domain to N1 and search list to N1,N2,
etc.
  root=NAME   - set root server to NAME
  retry=X     - set number of retries to X
  timeout=X   - set initial time-out interval to X seconds
  querytype=X - set query type, e.g., A,ANY,CNAME,HINFO,MX,NS,PTR,SOA,WKS
```

```
  type=X       - synonym for querytype
  class=X      - set query class to one of IN (Internet), CHAOS, HESIOD or
ANY
  [no]debug    - print debugging information
  [no]d2       - print exhaustive debugging information
exit           - exit the program, ^Z also exits
```

ls

Lists the information for a domain. The default output contains host names and their Internet addresses.

You can redirect the output from the `ls` command to a file as follows:

```
ls ... > filename.ext
ls ... >> filename.ext
```

The **>** places the output into *filename.ext*. The **>>** appends the output onto file *filename.ext*. When you direct output to a file, NSLOOKUP prints hash marks after every 50 records received from the server.

Format

> **ls** ***[-option [-option...]] domain***

Enter the command in lowercase or it is considered to be hostname "LS".

Parameters

-option

See the Options that follow for a list of available options. Always precede an option with a hyphen. Separate multiple options with spaces.

domain

Domain for which to list information.

Options

-a

Lists aliases of hosts in the domain. Synonym for `ls -t CNAME`.

-d

Lists all records for the domain. Synonym for `ls -t ANY`.

-h

Lists CPU and operating system information for the domain. Synonym for `ls -t HINFO`.

-s

Lists well-known services of hosts in the domain. Synonym for `ls -t WKS`.

-t *[type]*

Lists all records of the *type* specified by the `set type` command, as in Table 28-1.

Table 28-1 Record Types (-t *type*)

Value...	Means...
any	All records for the domain
a	Host internet address
afsdb	AFS database
axfr	Zone transfer
cname	Canonical name for an alias
hinfo	Host CPU and operating system type
isdn	ISDN information
mb	Mailbox
mg	Mail group
minfo	Mail information
mr	Mailbox rename
mx	Mail exchanger
ns	Name server for the named zone
ptr	Host name if query is Internet address, otherwise pointer to other information
rp	Responsible person
rt	Route through binding
soa	Domain "start-of-authority" information
txt	Text information
uinfo	User information
wks	Supported well-known services
x25	X.25 information

Examples

1 This example lists information for domain SIRIUS:

```
> ls nene.com
[sirius.nene.com]
```

```
 nene.com.                    server = sirius.nene.co

 sirius                          192.168.95.1
 nene.com.                    server = nic.near.net
 nic.near.net.                   192.52.71.4
```

2 This example lists all records for domain SIRIUS:

```
> ls -d nene.com
[sirius.nene.com]
 nene.com.    SOA   sirius.nene.com leary.sirius.nene.com
. (76 86400 1800 3600000 86400)
 nene.com.    NS    sirius.nene.com
```

3 This example appends the output in Example 2 onto an existing FOOBAR.TXT file:

```
> ls -d nene.com >> foobar.txt
[sirius.nene.com]
###
Received 165 records.
>
```

root

Changes the default server to the domain namespace root server.

The default host is `a.root-servers.net`.

You can change the name of the root server with the **`set root`** command.

Format

> **root**

Enter the command in lowercase or it is considered to be hostname "ROOT".

Synonym

> **lserver a.root-servers.net**

Example

This example lists the root server name and address.

```
> root
Default Server:  a.root-servers.net
Address:  198.41.0.4
```

server
lserver

Changes the default server to the specified host:

- The `server` command uses the current default server to look up information about the host.
- The `lserver` command uses the initial server to look up information about the host.

If NSLOOKUP cannot find an authoritative answer, it returns the names of servers that might have the answer.

Format

> server *host*
> lserver *host*

Enter the command in lowercase or it is considered to be hostname "SERVER" or "LSERVER".

Parameter

host

Domain name of the host.

Examples

1 This example changes the default server to GEMMA.

```
> server gemma
Default Server:  gemma.nene.com
Served by:
- NIC.NEAR.NET
          192.52.71.4
          NENE.COM
- BU.EDU
          128.197.27.7
          NENE.COM
```

2 This example changes the default server back to `sirius`.

```
> lserver sirius
Default Server:  sirius.nene.com
Address:  192.168.95.1
```

set

Changes state information that affects the lookups.

Format

> set *option*

Enter the command in lowercase or it is considered to be hostname "SET".

The *options* are listed as under the Options heading. The abbreviated form of the option is underlined.

Synonym

$ nslookup -*option [-option...]*

Options

all

Prints the current value of all keyword options and information about the current default server and host.

class=*value*

Changes the query class to one of the values in Table 28-2. The class specifies the protocol group of the information. The abbreviation is **`cl`**.

Table 28-2 set class Values

Value	Description
`in`	Internet class (default value)
`chaos`	Chaos class
`hesiod`	MIT Athena Hesiod class
`any`	Wildcard for any of the above

debug
nodebug (default)

Turns debug mode on (**`debug`**). Debug provides information about the packet sent to the server and the resulting answer.

d2
nod2 (default)

Turns exhaustive debug mode on (**`d2`**), which essentially displays all fields of every packet. If you turn exhaustive debug mode off again (**`nod2`**) , you remain in debug mode (see **`debug`**).

defname (default)
nodefname

Appends (`defname`) the default domain name to every single-component lookup, or disables this function (`nodefname`). You should probably leave the append function on in most cases.

domain=*name* (default=local domain)

Changes the default domain name to *name* and appends the default domain name to a lookup request, if `defname` and `search` are set. The domain search list contains the parents of the default domain if it has at least two components in its name.

ignoretc
noignoretc (default)

Ignores packet truncation errors (`ignoretc`) on output, or disables this function (`noignoretc`). In most cases, you would want to display these errors.

port=*number*

Changes the default TCP/UDP name server port to *number*.

querytype=*value*

Changes the type of information returned by a query to one of the values in Table 28-3. The default *value* is `a`. The synonym is `type`.

Table 28-3 set type Values

Value	Description
`any`	All records for the domain
`a`	Host internet address
`afsdb`	AFS database
`axfr`	Zone transfer
`cname`	Canonical name for an alias
`hinfo`	Host CPU and operating system type
`isdn`	ISDN information
`mb`	Mailbox
`mg`	Mail group
`minfo`	Mail information
`mr`	Mailbox rename
`mx`	Mail exchanger

Table 28-3 set type Values (Continued)

Value	Description
`ns`	Name server for the named zone
`ptr`	Host name if query is Internet address, otherwise pointer to other information
`rp`	Responsible person
`rt`	Route through binding
`soa`	Domain "start-of-authority" information
`txt`	Text information
`uinfo`	User information
`wks`	Supported well-known services
`x25`	X.25 information

retry=*number* (default=2)

Sets the number of retries to *number*. If NSLOOKUP does not receive a reply to a request within the amount of time specified by **`set timeout`**, it resends the request. The retry value controls how many times NSLOOKUP resends a request before giving up.

root=*host* (default=a.root.servers.net)

Changes the name of the root server to *host*. This changes the default root server when using the `root` command.

recurse (default)
norecurse

The name server recursively queries other servers if it does not have the information (**`recurse`**), or does not do this (**`norecurse`**). You would normally want recursive queries.

search (default)
nosearch

Searches for each name in parent domains of the current domain (**`search`**), or disables this function (**`nosearch`**). If the lookup request contains at least one period but does not end with a trailing period, **`search`** appends the domain names in the domain search list to the request until the server returns an answer.

srchlist=*name1 [/name2 /... /name6]* (default=local domain)

Sets up a search list by changing the default domain name to *name1* and the domain search list to *name1*, *name2*, and so on, up to six names, each separated by a slash (**`/`**). This command overrides the default domain name and search list of the **`set domain`** command.

timeout=*interval* (default=10)

Changes the timeout interval NSLOOKUP uses while waiting for a reply to the *interval* number of seconds. (See also **retry**.)

type=*value*

See **querytype**.

vc

novc (default)

Uses a TCP virtual circuit when sending requests to the server (**vc**), or uses UDP or allows NSLOOKUP to determine whether to use a virtual circuit based on the size of the request (**novc**). You would normally use **novc**.

Examples

1 This example sets the search list to NENE.COM and COM.

```
> set srchl=nene.com/com
```

2 This example sets the query type to ANY. When listing the host, all records appear.

```
> set q=any
> sirius
Server:  sirius.nene.com
Address:  192.168.95.1

sirius.nene.com   internet address = 192.168.95.1
sirius.nene.com   CPU = VAXstation 4000-90   OS = VMS V5.5-2
SET SRCHL=NENE.COM/COMP.PROTOCOLS.SMTP
```

spawn

Spawns a subprocess in order to execute a DCL command.

Note! You cannot SPAWN with CAPTIVE accounts.

Format

> spawn *[command]*

Enter the command in lowercase or it is considered to be hostname "SPAWN".

Synonym

> ! *[command]*

Parameter

command

Command to execute. If omitted, TCPware starts a DCL subprocess.

view

Sorts and lists previous **`ls`** command output redirected to a filename.

Format

> **view** ***filename***

Enter the command in lowercase or it is considered to be hostname "VIEW".

Parameter

filename

Filename to which you redirected output during an **`ls`** command.

Example

This example redirects a listing output to the FOOBAR.TXT file, and views the contents of the file.

```
> ls sirius > foobar.txt
[sirius.nene.com]
###
Received 165 records.
> view foobar.txt
[sirius.nene.com]
 nene.com.           server = sirius.nene.com
 sirius                 192.168.95.1
 nene.com.           server = nic.near.net
 nic.near.net.          192.52.71.4
```

NSLOOKUP Utility Error Messages

If the lookup request was not successful, NSLOOKUP prints an error message. Valid error messages are:

Timed-out	The server did not respond to a request after the amount of time specified in `timeout` and the number of retries specified in `retry`.
`Non-existent domain`	The host or domain name does not exist.
`No response from server`	The server machine does not run a name server.
`No records`	The server does not have resource records of the currently specified query type for the host, although the host name is valid.
`Connection Refused` or `Network Is Unreachable`	NSLOOKUP could not make a connection to the name or finger server. Common error with `finger` and `ls` requests.
`Server failure`	The name server found an internal inconsistency in its database and could not return a valid answer.
Refused	The name server refused to service the request.
`Format error`	The name server found that the request packet was not in the proper format.

PING

The PING utility tells you whether a host is up and whether you can reach it. You have a choice of two PING versions, PING, and PING_V2. The two version are slightly different, with PING displaying a bit more information. PING_V2 originated in TCPware Version 2.

The PING utility uses the ICMP echo and echo reply messages. To use the PING utility, you need BYPASS or SYSPRV privilege. Also, always run the PING utility from an account that has NETMBX privileges.

Using the Newer PING

Before using the newer version of the PING utility, enter the following:

```
$ PING :== $TCPWARE:PING
```

To "ping" a host, enter:

```
$ PING [-rv] host [data-size [npackets]]
```

- •**-r** – Do not route: do not use another gateway to reach the destination; the destination must be on a local network
- **-v** – Verbose mode, which displays information on an invalid response

- ***host*** – Hostname or internet address of the host to "ping"
- ***data-size*** – Size, in bytes, of the ICMP echo data
- ***npackets*** – Number of packets to send; if omitted, PING sends an infinite number of packets

To terminate or interrupt PING, enter **Ctrl/C**.

Example 28-1 shows a PING command and the resulting output. The **Ctrl/C** appears because the user interrupted the output with **Ctrl/C**.

Example 28-1 PING Example

```
NETCU> PING process-gw

PING process-gw.nene.com (192.168.95.126): 56 data bytes
64 bytes from 192.168.95.126: icmp_seq=0. time=10.ms
64 bytes from 192.168.95.126: icmp_seq=1. time=0.ms
64 bytes from 192.168.95.126: icmp_seq=2. time=0.ms
64 bytes from 192.168.95.126: icmp_seq=3. time=0.ms


Ctrl/C

----process-gw.nene.com PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/2/10
$
```

Using PING_V2

PING_V2 is from TCPware Version 2. Before using PING_V2, enter the following:

```
$ PING :== $TCPWARE:PING_V2
```

To "ping" a host using PING_V2, enter:

```
$ PING host [timeout [data-size [interval [display]]]]
```

host	Hostname or internet address of the host to "ping"
timeout	Command timeout time, in seconds (the default is 20 seconds)
data-size	Size, in bytes, of the ICMP echo data (the default is 128 bytes)
interval	Interval, in seconds, between the retransmission of the ECHO request if you do not receive a response (the default is 1 second)
display	**Y** to enable or **N** to disable the display of the packets (the default is **N**)

If the host responds, a message appears indicating that the host is alive. If the host fails to respond within the allowed time, a message appears indicating that the host did not answer. The number of packets sent, number of invalid packets received (other ICMP packets that were not responses to the ECHO request), and the elapsed time also appear.

PING also returns a status code so that you can use command procedures to test whether various hosts are available.

Example 28-2 shows a PING_V2 command and the resulting output.

Example 28-2 PING_V2 Example

```
PING process-gw

process-gw is alive
1 echo message sent (0 invalid responses) in 0.01 seconds.
```

QUOTED

The Quote-of-the-Day service (QUOTED) is a useful tool to check for connectivity. QUOTED is a TCP-based character generator service. As implemented, the QUOTED server listens for TCP connections on TCP port 17. Once you establish a connection, the service sends a short message. The service then throws away any data it receives and closes the connection.

It is your responsibility to provide the quote and define the TCPWARE_QUOTE logical.

There is no specific syntax for the quote. The quote can have a total of 512 characters and is limited to the following characters:

- ASCII printable
- Space
- Carriage return
- Line feed

You need to define a system logical, TCPWARE_QUOTE to specify the quote for the server. The TCPWARE_QUOTE logical name can be either a string or a filename that includes the quote text. Prefix a filename with the `@` sign and enclose the definition or filename in quotation marks.

You need SYSNAM or SYSPRV privileges to define the system-wide logical. The following examples show three different ways to define the TCPWARE_QUOTE logical:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_QUOTE "Quote-of-the-day"
$ DEFINE/SYSTEM/EXEC TCPWARE_QUOTE "@SYS$MANAGER:QUOTE.TXT"
$ DEFINE/SYSTEM/EXEC TCPWARE_QUOTE "Today’s quote is",-
_$ "@SYS$MANAGER:QUOTE.TXT"
```

To test QUOTED, TELNET to the host that has a defined quote-of-the-day, as follows:

TELNET *host* QUOTE OR **TELNET *host* 17**

(Specifying QUOTE or 17 is identical since the QUOTE server port number is 17.)

For example, the following command does a TELNET operation to BARTLETT’s quote-of-the-day server:

```
$ TELNET BARTLETT QUOTE
```

The system displays:

```
%TCPWARE_TELNET-I-TRYING, trying BARTLETT,quote (192.168.5.75,17) ...
%TCPWARE_TELNET-I-ESCCHR, escape (attention) character is "^
"Quote-of-the-day"
```

If the system logical is not defined, the system displays:

```
%TCPWARE_TELNET-I-TRYING, trying BARTLETT,quote (192.168.5.75,17) ...
%TCPWARE_TELNET-I-ESCCHR, escape (attention) character is "^
No quote-of-the-day is currently defined.
```

TCPDUMP

TCPDUMP is a useful mechanism for tracking TCP packets by displaying information in the packet headers. You can specify the type of packet information to extract by including the relevant options and expressions.

Use of TCPDUMP assumes a thorough understanding of the TCP protocol.

Before using TCPDUMP, enter the following foreign command definition:

```
$ TCPDUMP:==$TCPWARE:TCPDUMP
```

The TCPDUMP command syntax is as follows:

```
$ TCPDUMP [options] [expressions]
```

You can also use the TCPDUMP command on the Network Control Utility (NETCU) level. This allows you to use OpenVMS qualifiers in place of (or in addition to) UNIX-style options. To use TCPDUMP on the NETCU level, enter the following:

```
$ NETCU TCPDUMP [qualifiers | options] [expressions]
```

For complete details on the available options and expressions, see *TCPDUMP Command Reference*.

Interpreting TCPDUMP Output

Much of the information that follows is adapted from material provided by the Lawrence Berkeley Laboratory, University of California, Berkeley, and must contain the following notice:

The output of TCPDUMP is protocol-dependent. The following gives a brief description and examples of most of the various output formats. The description assumes familiarity with RFC 793, *Transmission Control Protocol*.

Monitoring TCP Packets

Consider the command and its output in Example 28-3.

Example 28-3 TCPDUMP Output

```
$ tcpdump host bart
Getting stats.
tcpdump: listening on ESA0:
18:21:59.000400 bart.nene.com.1023 > marge.login: S
560913442:560913442(0) win 24576 <mss 1460,wscale 0,eol> (DF)
18:21:59.004000 bart.nene.com.1023 > marge.login: . ack 513152385
```

```
win 24576 (DF)
18:21:59.004100 bart.nene.com.1023 > marge.login: P 0:1(1) ack 1 win 24576
(DF)
```

The general format of a TCP protocol line is as follows, with references to Example 28-3 in the explanations given in parentheses:

timestamp src > dst*: *flags [first-byte*:*last-byte* (*total-bytes*) ack *seqno
win *bytes* urg *<options>]*

timestamp (`18:21:59.000400`)	By default, all output lines are preceded by a time stamp, the current clock time in the form *hours*:*minutes*:*seconds*.*decimal*. The time stamp reflects the time the driver delivers the packet. No attempt is made to account for the time lag between when the Ethernet interface removed the packet from the wire and when the driver services the I/O request. The additional **`-tt`** option you can use with the TCPDUMP command displays an unformatted time stamp, while the **`-t`** option removes the time stamp altogether.
***src* > *dst*:** (`bart.nene.com.1023 > marge.login:`)	Source and destination IP address and port.
flags – **`S`** (SYN), **`F`** (FIN), **`P`** (PUSH), **`R`** (RST), or **`.`** (no flag)	Transition flags that can appear in combination.
***first-byte*: *last-byte* (*total-bytes*)** (`560913442:560913442(0)`)	Packet data sequence number, the first byte followed by the last byte of data, and the total bytes.
ack *seqno* (`ack 513152385`)	Sequence number of the next data expected in the other direction on this connection.
win *bytes* (`win 24576`)	Number of bytes of receive buffer space available in the other direction on this connection.
urg (not shown)	The packet contains "Urgent" data.
<*options*> (not shown)	TCP options, enclosed in angle brackets (**< >**).
(DF) (various lines)	The IP do-not-fragment flag is included with the packet.

The ***src*** and ***dst*** values and the flags (**S**, **F**, **P**, **R**, or **.**) are always present. The other fields depend on the contents of the packet's TCP protocol header and appear only if appropriate.

Example 28-4 shows the opening portion of an RLOGIN operation from BART to MARGE.

Example 28-4 RLOGIN Output

```
bart.1023 > marge.login: S 768512:768512(0) win 4096 <mss 1024>
marge.login > bart.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
bart.1023 > marge.login: . ack 1 win 4096
bart.1023 > marge.login: P 1:2(1) ack 1 win 4096
marge.login > bart.1023: . ack 2 win 4096
bart.1023 > marge.login: P 2:21(19) ack 1 win 4096
marge.login > bart.1023: P 1:2(1) ack 21 win 4077
marge.login > bart.1023: P 2:3(1) ack 21 win 4077 urg 1
marge.login > bart.1023: P 3:4(1) ack 21 win 4077 urg 1
```

Here is the explanation for each line in Example 54-4:

- **`bart.1023 > marge.login: S 768512:768512(0) win 4096 <mss 1024>`**

  TCP port 1023 on BART sends a packet to port LOGIN on MARGE. The `S` indicates that the SYN flag was set. The packet sequence number is 768512 and it contains no data. There is no piggybacked ACK, the available receive window is 4096 bytes, and there is a maximum segment size (MSS) option requesting an MSS of 1024 bytes.

- **`marge.login > bart.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>`**

  MARGE replies with a similar packet except that it includes a piggy-backed ACK in response to BART's SYN.

- **`bart.1023 > marge.login: . ack 1 win 4096`**

  BART then sends an ACK in response to MARGE's SYN. The **`.`** means that no flags were set. The packet contains no data, so there is no data sequence number. Note that the ACK sequence number is a small integer (1). The first time TCPDUMP sees a TCP "conversation," it displays the sequence number from the packet.

- **`bart.1023 > marge.login: P 1:2(1) ack 1 win 4096`**

  marge.login > bart.1023: . ack 2 win 4096

  On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is displayed. This means that sequence numbers after the initial one can be interpreted as relative byte positions in the conversation's data stream (with the first data byte in each direction being `1`). (The **`-S`** option would override this feature, causing the original sequence numbers to be output.)

- **`bart.1023 > marge.login: P 2:21(19) ack 1 win 4096`**

  BART sends MARGE 19 bytes of data (bytes 2 through 20 in the BART-to-MARGE side of the conversation). The **`P`** (PUSH) flag is set in the packet.

- **`marge.login > bart.1023: P 1:2(1) ack 21 win 4077`**

  MARGE indicates that it received data up to, but not including, byte 21. Most of this data is apparently sitting in the socket buffer, since MARGE's receive window decreased by 19 bytes. MARGE also sends one byte of data to BART in this packet.

- **marge.login > bart.1023: P 2:3(1) ack 21 win 4077 urg 1**

  marge.login > bart.1023: P 3:4(1) ack 21 win 4077 urg 1

  MARGE sends two bytes of urgent, pushed data to BART.

Monitoring UDP Packets

The UDP format is illustrated by this **rwho** packet:

```
molly.who > broadcast.who: udp 84
```

This says that port WHO on host MOLLY sent a UDP datagram to port WHO on host BROADCAST, the Internet broadcast address. The packet contained 84 bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information is displayed; in particular, Domain Name System (DNS) service requests and Sun Remote Procedure Calls to the Network File System (NFS).

Displaying Link Level Headers

The **-e** option of the TCPDUMP command displays the link level header of each dump line. On Ethernet systems, this displays the source and destination addresses, protocol, and packet length. For example, the following command displays the additional information shown in **bold** type:

```
$ tcpdump -e host bart -t
Getting stats.
tcpdump: listening on ESA0:
aa:0:4:0:3a:8 aa:0:4:0:15:a ip 62: bart.nene.com.1023 > marge.login: S
176494692:176494692(0) win 24576 <mss 1460,wscale 0,eol> (DF)
aa:0:4:0:3a:8 aa:0:4:0:15:a ip 60: bart.nene.com.1023 > marge.login: . ack
128739885 win 24576 (DF)
aa:0:4:0:3a:8 aa:0:4:0:15:a ip 60: bart.nene.com.1023 > marge.login: P
0:1(1) ack 1 win 24576 (DF)
```

The link level headers begin with **aa**. (Note that time stamps were omitted since the **-t** option was used.)

On FDDI networks, the **-e** option displays the frame control field – the source and destination addresses, and the packet length – that governs the interpretation of the rest of the packet.

Normal packets (such as those containing IP datagrams) are asynchronous, with a priority value between 0 and 7; for example, `async4`. Such packets are assumed to contain an 802.2 Logical Link Control (LLC) packet; the LLC header is displayed if it is not an ISO datagram or so-called Subnetwork Access Protocol (SNAP) packet.

Monitoring ARP and RARP Packets

The following description assumes familiarity with the ARP and RARP protocols. See Chapter 6, *Common Interfaces*, the *Address Resolution Protocol (ARP)* and *Reverse Address Resolution Protocol (RARP)* subsections for further information.

Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP) output shows the type of request and its arguments.

The format is intended to be self-explanatory. The following abbreviated sample output is taken from the start of an RLOGIN operation from host MARGE to BART:

```
$ tcpdump host bart -t
arp who-has bart.process.com tell marge.nene.com
arp reply bart.nene.com is-at aa:0:4:0:1f:8
```

The first line indicates that MARGE sent an ARP packet asking for BART's Ethernet address. BART replied with its Ethernet address, aa:0:4:0:1f:8.

You can also specify not to convert IP addresses to hostnames by using the **-n** option:

```
$ tcpdump host bart -t -n
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at aa:0:4:0:1f:8
```

With the additional link level header information, the output would be as follows:

```
$ tcpdump host bart -t -e
aa:0:4:0:1:8 aa:0:4:0:15:a ip 82: arp who-has 128.3.254.6 tell
128.3.254.68
aa:0:4:0:1:8 aa:0:4:0:15:a ip 82: arp reply 128.3.254.6 is-at
aa:0:4:0:1f:8
```

DNS Name Server Requests

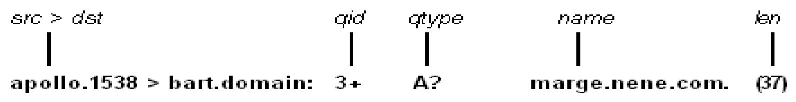
The following description assumes familiarity with the DNS protocol. See Chapter 1, *Domain Name Services*, for further information.

Domain Name System (DNS) name server requests are formatted as:

***src* > *dst*: *qid op*? *flags qtype qclass name* (*len*)**

Figure 28-1 shows an example in which APOLLO queries the domain server on BART for an address (type A) record associated with the name MARGE.NENE.COM.

Figure 28-1 DNS Name Server Request Display



- The source and destination hosts (***src > dst***) are given.
- The query id (***qid***) is **3** (the **+** indicates that the "recursion desired" flag was set).
- The query operation (***op*?**) is omitted since it is the normal one, Query. If *op* had been anything else, it would appear between the 3 and the +.

- The query type (***qtype***) is **A** for Address record.
- The ***qclass*** is omitted since it is the normal one, C_IN. Any other *qclass* would have been displayed immediately after the A.
- The ***name*** is the domain name of target host MARGE.
- The query length (***len***) is **37** bytes, not including the UDP and IP protocol headers.

A few anomalies are checked and can result in extra fields enclosed in square brackets. If a query contains an answer, name server, or authority section, *ancount*, *nscount*, or *arcount* are displayed as **[*n*a]**, **[*n*n]**, or **[*n*au]**, where *n* is the appropriate count. If any of the response bits are set (AA, RA, or *rcode*) or any of the "must be zero" bits are set in bytes two and three, **[b2&3=*x*]** is displayed, where *x* is the hex value of header bytes two and three.

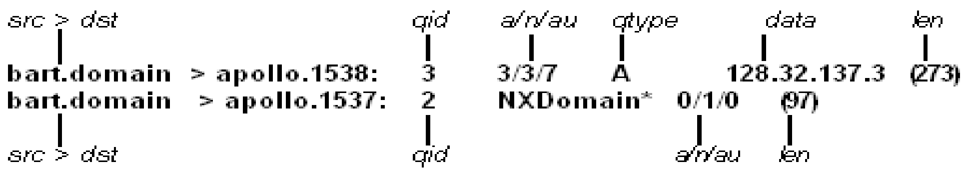
DNS Name Server Responses

DNS name server responses are formatted as:

src > dst: qid op rcode flags a/n/au qtype qclass data (len)

Figure 28-2 shows two examples.

Figure 28-2 DNS Name Server Response Display



In the first example, BART responds to query 3 from APOLLO with three answer records (*a*), three name server records (*n*), and seven authority records (*au*). The first answer record is type A (address) and its data is internet address 128.32.137.3. The total size of the response is 273 bytes, excluding UDP and IP headers. The *op* (Query) and *rcode* (NoError) were omitted, as was the *class* (C_IN) of the A record.

In the second example, BART responds to query 2 with a response code of nonexistent domain (**NXDomain**) with no answers, one name server, and no authority records. The ***** indicates that the authoritative answer bit was set. Since there were no answers, there is no *qtype*, *qclass*, or *data* displayed.

Other flag characters that might appear are **-** (recursion available, **RA**, not set) and **|** (truncated message, **TC**, set). If the "question" section does not contain exactly one entry, **[nq]** is displayed.

Note that name server requests and responses tend to be large, and the default snapshot length of 68 bytes may not capture enough of the packet to display. Use the **-s** flag to increase the snapshot length if you need to seriously investigate name server traffic (**-s 128** is known to be effective).

NFS Requests and Replies

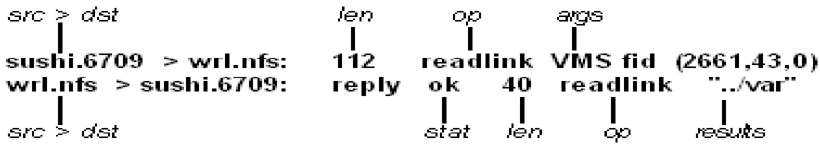
The following description assumes familiarity with the Network File System (NFS). See Chapter 12, *NFS-OpenVMS Client Management*, or Chapter 13, *NFS-OpenVMS Server Management*, for further information.

NFS requests and replies are displayed as:

src.xid* > *dst*.nfs: *len op args
src*.nfs > *dst.xid*: reply *stat len op results

Figure 28-3 shows an example. SUSHI sends a transaction with *id* 6709 to WRL (note that the number following the *src* host is the transaction ID, not the source port). The request was 112 bytes long, excluding the UDP and IP headers. The operation was a **`readlink`** (read symbolic link) on file handle (**`VMS fid`**) **`2661,43,0`**. Host WRL replies **`ok`** with the contents of the link.

Figure 28-3 NFS Requests and Replies Display



If you use the **`-v`** (verbose) option, additional information is displayed. For example:

```
sushi.1372a > wrl.nfs: 148 read VMS fid (2661,43,0) 8192 bytes @ 24576
wrl.nfs > sushi.1372a: reply ok 1472 read REG 100664 ids 417/0 sz 29388
```

In the first line, SUSHI queries WRL to read 8192 bytes from file handle 2661,43,0 at byte offset **`@ 24576`**. WRL responds with **`ok`**. The packet shown on the second line is the first fragment of the reply, and hence is only 1472 bytes long. (The other bytes follow in subsequent fragments, but these fragments do not have NFS or even UDP headers and so might not be displayed, depending on the filter expression used.)

Because of the **`-v`** option, some of the file attributes returned in addition to the file data are displayed: the file type (**`REG`**, for regular file), the file mode (in octal, **`100664`**), the UID and GID (**`417/0`**), and the file size (**`sz`**, **`29388`**) . (The **`-v`** option also displays the IP header TTL, ID, and fragmentation fields, which were omitted from this example.) If you use **`-v`** more than once, even more details are displayed.

Note that NFS requests are very large and much of the detail is not displayed unless the snapshot length is increased. Try using **`-s 192`** to watch NFS traffic.

NFS reply packets do not explicitly identify the RPC operation. Instead, TCPDUMP keeps track of "recent" requests, and matches them to the responses using the transaction ID. If a response does not closely follow the corresponding request, it might be ignored.

TCPDUMP supports the following options/qualifiers enabling more decoding of RPC-based services.

```
R_RPC

-"R" all|udp|tcp
/RPC [=ALL|UDP|TCP]
```

For a UNIX-style option, the “R” must be uppercase and quoted.

The option/qualifer values are

ALL (default)	Decode both UDP and TCP
UDP	Decode only UDP
TCP	Decode only TCP

The following RPC protocols are decoded:

NFS	Portmapper
Network lock manager version 1	Network lock manager version 3
Net status	Mount
PCNFS version 1	PCNFS version 2

IP Fragmentation

Fragmented Internet datagrams are displayed as:

(frag *id*:*size@offset*+)
(frag *id*:*size@offset*)

The first form, which includes the ending **+**, indicates that there are more fragments. The second form indicates the last fragment. The ***id*** parameter is the fragment ID, ***size*** is the fragment size (in bytes) excluding the IP header, and ***offset*** is this fragment's offset (in bytes) in the original datagram.

Information is displayed for each fragment. The first fragment contains the higher level protocol header and the fragment information is displayed after the protocol information. Fragments after the first contain no higher level protocol header, and the fragment information is displayed after the source and destination addresses. For example, here is part of an FTP from NENE.EDU to LBL-BART.ARPA over a CSNET connection that does not appear to handle 576-byte datagrams:

```
nene.ftp-data > bart.1170: . 1024:1332(308) ack 1 win 4096 (frag
595a:328@0+)
nene > bart: (frag 595a:204@328)
bart.1170 > nene.ftp-data: . ack 1536 win 2560
```

- Addresses in the third line do not include port numbers, because the TCP protocol information is all in the first fragment, and it is not know what the port or sequence numbers will be when the later fragments are displayed.
- The TCP sequence information in the first line is displayed as if there were 308 bytes of user data when, in fact, there are 512 bytes (308 in the first fragment and 204 in the second). If you are looking for holes in the sequence space or trying to match up ACKs with packets, this can be misleading.

A packet with the IP do-not-fragment flag is marked with a trailing **(DF)**.

TCPDUMP Command Reference

This section shows the format for and examples of the TCPDUMP command.

The first form of the TCPDUMP command is available on the DCL level as:

```
$ TCPDUMP:==$TCPWARE:TCPDUMP
$ TCPDUMP [options] [expressions]
```

The second form is on the Network Control Utility (NETCU) level. This form allows you to use OpenVMS qualifiers in place of (or in addition to) UNIX-style options. To use TCPDUMP on the NETCU level, enter the following:

```
$ NETCU TCPDUMP [qualifiers | options] [expressions]
```

The options and their qualifier equivalents are listed together.

TCPDUMP

TCPDUMP displays the headers of packets on a network interface that match the boolean expression. The OpenVMS implementation currently works only with Compaq-compatible Ethernet cards. Some of the command line switches were changed from the UNIX version to support OpenVMS's case-insensitive command line.

PHY_IO privilege is required to use TCPDUMP, unless reading packets from a file. If using the TCPware drivers for packet capturing, LOG_IO and SYSPRV or BYPASS privileges are also needed.

Format

TCPDUMP *[options | qualifiers] [expressions]*

Options and Qualifiers

Note! Only options are available if using TCPDUMP as a foreign command at the DCL prompt. You can mix and match options and qualifiers if using TCPDUMP as a NETCU command. In each case, the option is listed before its equivalent qualifier.

-a
/NO_RELATIVE_SEQUENCE_NUMBERS

Displays absolute, rather than relative, TCP sequence numbers. /RELATIVE is the default.

-b *bufcount*
/BUFFERS=*bufcount*

Sets the number of receive buffers for the Ethernet adapter to *bufcount*. The default is 255 on VAX, 175 on AXP. You may need to lower the number if you receive the message `%SYSTEM-F-EXQUOTA, process quota exceeded`. This option does not apply to the TCPware drivers. Valid values are 1 to 255.

-c *exitcount*
/COUNT=*exitcount*

Exits after receiving *exitcount* packets.

-d
/HOSTNAMES=NOQUALIFIED

Does not display host names as fully qualified domain names. For example, display **`nic`** instead of **`nic.ddn.mil`**. /HOSTNAMES=QUALIFIED is the default.

-e
/LINK_HEADERS

Displays the link-level header on each dump line. (See *Displaying Link Level Headers*.)

-f
/HOSTNAMES=NOFOREIGN

Displays "foreign" internet addresses numerically, not symbolically. FOREIGN is the default.

-g
/NETWORK

Translates network and broadcast addresses to domain names.

-i *interface*
/INTERFACE=*interface*

Listens on the *interface* interface. If unspecified, TCPDUMP searches for a configured interface (excluding loopback). Supported interfaces appear in Table 28-4.

Table 28-4 Supported Interfaces for TCPDUMP

Device...	Which is...	On platform...	For medium...
UVA-0	DELUA, DEUNA	VAX, AXP	Ethernet
QNA-0	DEQNA, DELQA, DEQTA	VAX, AXP	Ethernet
BNA-0	DEBNA, DEBNI	VAX, AXP	Ethernet
SVA-0	DESVA	VAX, AXP	Ethernet
MNA-0	DEMNA	VAX, AXP	Ethernet
ISA-0	SGEC, TGEC	VAX, AXP	Ethernet
MXE-0	PMAD	VAX, AXP	Ethernet
ERA-0	DE422, DE425	AXP	Ethernet
EWA-0	TULIP	AXP	Ethernet
MFA-0	DEMFA	VAX, AXP	FDDI
FZA-0	DEFZA, DEFTA	VAX, AXP	FDDI
FAA-0	DEFAA	VAX, AXP	FDDI
FEA-0	DEFEA	AXP	FDDI
FQA-0	DEFQA	VAX, AXP	FDDI
FPA-0	DEFPA	AXP	FDDI
IP	IPDRIVER	VAX, AXP	N/A *****
UDP	UDPDRIVER	VAX, AXP	N/A *****

Table 28-4 Supported Interfaces for TCPDUMP (Continued)

TCP	TCPDRIVER	VAX, AXP	N/A *
* Because these drivers are device-independent, filter expressions based on portions of the Ethernet header are not valid. Additionally, only packets valid for that particular driver are delivered to TCPDUMP (UDPDRIVER only delivers UDP packets, not TCP or other protocols such as ARP/RARP). When using higher level drivers to capture packets, it is also not necessary to contain the protocol in the filter expression: **`TCPDUMP -i udp udp and host marge`** is redundant and results in additional overhead while processing packets. The proper usage would be **`TCPDUMP -i udp host marge`**. BOOTP requests are displayed, though with an invalid source address. The advantage of using these drivers is the ability to view your own packets, as well as a decrease in the number of lost packets.			

-j *file*
/FILTER_EXPRESSIONS=*file*

Uses the specified file as input for the filter expression. Any additional expressions given on the command line are ignored.

-k *packetype*
/PACKET_TYPE=*packetype*

Forces packets selected by the expression to be interpreted by the specified *packetype*. Currently known types are **`rpc`** (Remote Procedure Call), **`rtp`** (Real-Time Applications Protocol), **`rtcp`** (Real-Time Applications Control Protocol), **`vat`** (Visual Audio Tool), and **`wb`** (Distributed White Board).

-l
/LINE_BUFFERED

Makes `stdout` line buffered.

-n
/NOHOSTNAMES

Does not convert addresses (host addresses, port numbers, and so on) to names. (See *Monitoring ARP and RARP Packets* for an example.)

-o
/NOOPTIMIZER

Does not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer. The default is /OPTIMIZER.

-q
/SHOW=LESS

Quick output. Displays less protocol information so that output lines are shorter. The default is

/SHOW=NORMAL.

-r *file*
/INPUT=*file*

Reads packets from the file created with the **-w** option.

-s *snapshotlength*
/DATA=*snapshotlength*

Captures *snapshotlength* bytes of data from each packet, rather than the default of 68 bytes. 68 bytes is adequate for IP, ICMP, TCP, and UDP, but may truncate protocol information from name server and NFS packets. Packets truncated because of a limited snapshot are indicated in the output with **[|*proto*]**, where *proto* is the name of the protocol level at which the truncation occurred.

If a snapshot of IP data is smaller than the actual packet size, the message **`truncated-ip - n bytes missing!`** appears. This is an informational message, and the data captured may be sufficient. If you need to see more data, use **`-s`** (or /DATA) to increase the snapshot length.

Note that taking larger snapshots both increases the amount of time it takes to process packets, and effectively decreases the amount of packet buffering. This can cause packets to be lost. You should limit *snapshotlength* to the smallest number that captures the protocol information you need.

-t
/NOTIME

Does not display a time stamp on each dump line. The default is /TIME=FORMAT.

-tt
/TIME=NOFORMAT

Displays an unformatted time stamp on each dump line. The default is /TIME=FORMAT.

-v
/SHOW=MORE

Verbose output. For example, the time-to-live (TTL) and type-of-service (TOS) information in an IP packet is displayed. (See *NFS Requests and Replies*.)

-vv
/SHOW=FULL

"Hyperverbose" output. For example, additional fields are displayed from NFS reply packets.

-w *file*
/OUTPUT=*file*

Writes the raw packets to a file, rather than parsing and displaying them out. The file can later be displayed with the **`-r`** option.

-x
/HEXADECIMAL

Displays each packet (minus its link level header) in hexadecimal format. The smaller of the entire packet and *snapshotlength* byte values are displayed.

-z
/ASCII

Displays each packet in hexadecimal format with the ASCII equivalent.

Expressions

Recall that a TCPDUMP command consists of the following elements:

TCPDUMP *[options* **|** *qualifiers] [* ***expressions*** *]*

Expressions select which packets to dump. Only packets for which the expression is true are dumped (or all packets if the expression is omitted). An expression breaks down into one or more primitives:

expression = primitive primitive ...

Primitives usually break down into a qualifier and an address:

primitive = qualifier address

Expression Qualifiers

There are three kinds of qualifiers – type, direction, and protocol:

- ***Type*** qualifiers define the ID name or number type. Possible types are:
  - **`host`** (default)
  - **`net`**
  - **`port`**

  Examples of primitives using these qualifiers are **`host foo`**, **`net 128.3`**, and **`port 20`**. The default is **`host`** if the type qualifier is omitted, so that **`foo`** and **`host foo`** are equivalent. A host can be either an IP address or hostname. A **`net`** can either be a name from the `TCPWARE:NETWORKS.` file, or a network number, and can include the additional **`mask`** qualifier.
- ***Direction*** qualifiers specify a transfer direction, and are prepended to the *type* qualifier:
  - **`src`**
  - **`dst`**
  - **`src or dst`** `(default)`
  - **`src and dst`**

  Examples of primitives using these qualifiers are **`src foo`**, **`dst net 128.3`**, and **`src or dst port ftp-data`**. The default is **`src or dst`** if the direction qualifier is omitted, so that **`port ftp-data`** is true for a source or destination port.
- ***Protocol*** qualifiers restrict the match to a particular protocol, and are prepended to the *direction* qualifier. Possible protocols are:
  - **`ether`**
  - **`fddi`**

- **`icmp`**
- **`ip`**
- **`ipx`**
- **`arp`**
- **`rarp`**
- **`decnet`**
- **`lat`**
- **`sca`**
- **`moprc`**
- **`mopdl`**
- **`tcp`**
- **`udp`**

Examples of primitives using these qualifiers are **`ether src foo`**, **`arp net 128.3`**, and **`tcp port 21`**. If the protocol qualifier is omitted, the default is all protocols consistent with the type. For example:

- **`src foo`** implies **`ip`**, **`arp`**, or **`rarp`**
- **`net bar`** implies **`ip`**, **`arp`**, or **`rarp`**
- **`port 53`** implies **`tcp`** or **`udp`**

You can be even more specific about the **`ip`** protocol by adding **`proto`** and the IP protocols **`cmp`**, **`igrp`**, **`dp`**, **`nd`**, or **`cp`**; for example, **`ip proto cmp`**. (Note that some protocols are preceded by backslashes to distinguish them from protocol qualifiers.) The same goes for the **`ether`** protocols **`p`**, **`rp`**, and **`arp`**, such as in **`ether proto p`**.

The **`fddi`** protocol qualifier is actually an alias for **`ether`**, and are treated identically as meaning the data link level used on the specified network interface. FDDI headers contain Ethernet-like source and destination addresses, and often contain Ethernet-like packet types, so you can filter on these FDDI fields just as with the analogous Ethernet fields. (FDDI headers also contain other fields, but you cannot name them explicitly in a filter expression.)

Other Expressions

There are additional special, "primitive" qualifiers that do not follow the pattern:

- **`gateway`**
- **`broadcast`**
- **`less`**
- **`greater`**
- arithmetic expressions

All of these are described in the *Primitives* subsection that follows.

More complex filter expressions are built up by using the words **`and`**, **`or`**, and **`not`** to combine primitives. For example, **`host foo and not port ftp and not port ftp-data`**. To save typing, identical qualifier lists can be omitted. For example, the following two expressions are identical, the first being the abbreviated form:

tcp dst port ftp or ftp-data or domain
tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain.

Expression Primitives

dst *[***host***]* *host*

IP destination field of the packet must be *host*.

src *[***host***]* *host*

IP source field of the packet must be *host*.

*[***host***]* *host*

IP source or destination of the packet must be *host*. Any of the above host expressions can be prepended with the keywords **ip**, **arp**, or **rarp**, as in **ip host *host***, which is equivalent to **ether proto ip and host** *host*. If *host* is a name with multiple IP addresses, each address is checked for a match.

ether dst *ehost*

Ethernet destination address must be *ehost*, which is either a name from the TCPWARE:ETHERS. file, or a number.

ether src *ehost*

Ethernet source address must be *ehost*.

ether *[***host***]* *ehost*

Ethernet source or destination address must be *ehost.*

gateway *host*

Packet must use *host* as a gateway; that is, the Ethernet source or destination address was *host*, but neither the IP source nor destination was *host*. (This makes the expression equivalent to **ether host *ehost* and not host *host***.) The *host* must be a name and must be found in both the TCPWARE:HOSTS. and TCPWARE:ETHERS. files.

dst net *net*

IP destination address of the packet must have a network ID of *net*.

src net *net*

IP source address of the packet must have a network ID of *net*.

net *net*

Either the IP source or destination address of the packet must have a network ID of *net.*

net *net* **mask** *mask*

IP address must match its network ID with the specified *mask*; can be qualified with **src** or **dst**.

net *net***/***length*

IP address must match its network ID with the specified mask length, in bits; can be qualified with **src** or **dst**.

dst port *port*

Packet must be IP/TCP or IP/UDP and have a destination port value, a number or name used in the `TCPWARE:SERVICES.` file. If a name, both the port number and protocol are checked. If a number or ambiguous name, only the port number is checked. For example, **dst port 513** displays both TCP/LOGIN and UDP/WHO traffic; **dst port domain** displays both TCP/DOMAIN and UDP/DOMAIN traffic.

src port *port*

Packet must have a source port value.

port *port*

Either the source or destination port of the packet must be *port*. Any of the above port expressions can be prepended with the keywords **tcp** or **udp**, as in **tcp src port port**, which matches only TCP packets whose source port is *port*.

less *length*

Packet must have a length less than or equal to *length* (equivalent to **len <= *length***).

greater *length*

Packet must have a length greater than or equal to *length* (equivalent to **len >= *length***).

ip proto *protocol*

Packet must be an IP packet of protocol type *protocol*, which can be a number or one of the names **cmp**, **igrp**, **dp**, **nd**, or **cp**. Note that **tcp**, **udp**, and **icmp** must be escaped using a backslash (`F0>`) `to distinguish them from qualifiers.`

*[***ether***]* **broadcast**

Packet must be an Ethernet broadcast packet. The **ether** keyword is optional.

ip broadcast

Packet must be an IP broadcast packet. It checks for both the all-zeros and all-ones broadcast conventions, and looks up the local subnet mask.

*[***ether***]* **multicast**

Packet must be an Ethernet multicast packet (shorthand for **ether[0] & 1 != 0**).

ip multicast

Packet must be an IP multicast packet.

ether proto *protocol*

Packet must be of Ethernet type *protocol*, which can be a number or name like **p**, **rp**, or F30@Z7@Lam>rarp. Note that these identifiers must be escaped using a backslash (\)**.**

In the case of FDDI (such as **fddi proto arp**), the protocol identification comes from the 802.2 Logical Link Control (LLC) header, which is usually layered on top of the FDDI header. TCPDUMP assumes, when filtering on the protocol identifier, that all FDDI packets include an LLC header, and that the LLC header is in so-called Subnetwork Access Protocol (SNAP) format.

decnet src *host*

DECNET source address must be *host*, which can be an address of the form 10.123, or a DECNET hostname.

decnet dst *host*

DECNET destination address must be *host.*

decnet host *host*

Either the DECNET source or destination address must be *host*.

ip, arp, rarp, decnet

Abbreviation for **ether proto *protocol*** for one of these protocols.

ipx { **src** | **dst** } *host*

With **ipx** alone, packet must be an IPX packet. With **src** or **dst**, the packet must come from or go to an ethernet host.

lat, moprc, mopdl

Abbreviation for **ether proto *protocol*** for one of these protocols. Note that TCPDUMP does not currently know how to parse these protocols.

tcp, udp, icmp

Abbreviation for **ether proto *protocol*** from one of these protocols.

expr *relational-operator expression*

The relation must hold, where *relational-operator* is one of **>**, **<**, **>=**, **<=**, **=**, **!=**, and *expression* is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators (**+**, **-**, *****, **/**, **&**, **|**), a length operator, and special packet data accessors (see the following subsection).

Accessing Data Within a Packet

To access data inside a packet, use the syntax:

***proto* [*expr* : *size*]**

- ***proto*** is one of **ether**, **fddi**, **ip**, **arp**, **rarp**, **tcp**, **udp**, or **icmp**, and indicates the protocol layer for the index operation.
- ***expr*** is the byte offset, relative to the indicated protocol layer.
- ***size*** is optional and indicates the number of bytes in the field of interest. It can be either **1**, **2**, or **4**, and defaults to **1**. The length operator, indicated by the keyword **len**, gives the length of the packet.

For example:

- **ether[0] & 1 != 0** catches all multicast traffic.
- **ip[0] & 0xf != 5** catches all IP packets with options.
- **ip[6:2] & 0x1fff = 0** catches only unfragmented datagrams and fragment zero of fragmented datagrams. This check is implicitly applied to the TCP and UDP index operations. For instance, **tcp[0]** always means the first byte of the TCP header, and never means the first byte of an intervening fragment.

Combining Primitives

You can combine primitives using:

- A parenthesized group of primitives and operators (parentheses must be escaped)
- Negation (**!** or **not**) (you must enclose expressions using **!** in quotes)
- Concatenation (**&** or **and**)
- Alternation (**|** or **or**)

Negation has the highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit **and** tokens, not juxtaposition, are required for concatenation.

If the identifier omits a qualifier, the most recent qualifier applies. For example,

not host vs and ace is short for **not host vs and host ace**, which should not be confused with **not (host vs or ace)**.

Expression arguments can be passed to TCPDUMP as either a single argument or as multiple arguments, whichever is more convenient.

Examples

1. These identical examples display all packets arriving at or departing from BART.
   ```
   $ tcpdump bart
   $ tcpdump host bart
   ```
2. These identical examples display traffic between HELIOS and either HOT or ACE.
   ```
   $ tcpdump helios and (hot or ace)
   ```

```
$ tcpdump host helios and (hot or ace)
```

3 These identical examples display all IP packets between ACE and any host except HELOIS.

```
$ tcpdump ip ace and not helios
$ tcpdump ip host ace and not helios
```

4 This example displays all traffic between local hosts and hosts at Berkeley.

```
$ tcpdump net ucb-ether
```

5 This example displays all FTP traffic through gateway SNUP (note that the expression is quoted because of the parentheses).

```
$ tcpdump "gateway snup and (port ftp or ftp-data)"
```

6 This example displays traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this information should never make it onto your local net).

```
$ tcpdump ip and not net localnet
```

7 This example displays the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a nonlocal host.

```
$ tcpdump "tcp[13] & 3 != 0 and not src and dst net localnet"
```

8 This example displays IP packets longer than 576 bytes sent through gateway HOMER.

```
$ tcpdump "gateway homer and ip[2:2] > 576"
```

9 This example displays IP broadcast or multicast packets not sent over Ethernet broadcast or multicast.

```
$ tcpdump "ether[0] & 1 = 0 and ip[16] >= 224"
```

10 This example displays all ICMP packets that are not echo requests/replies (PING packets).

```
$ tcpdump "icmp[0] != 8 and icmp[0] != 0"
```

TIME

TIME is the Time Protocol, defined by RFC 868. For a TCP connection, TIME returns a four-byte integer (in network byte order) of the current time in number of seconds since 1-JAN-1900 00:00 GMT. For UDP, when you receive a packet time port (37), TIME returns a four-byte datagram with the time (and discards the received datagram).

TRACEROUTE

This section describes the TRACEROUTE utility, which traces the path of an IP packet to an internet host. Only one user can run TRACEROUTE at a time. TRACEROUTE requires BYPASS or SYSPRV privileges.

Before using TRACEROUTE, enter the following to define the TRACEROUTE foreign command definition:

```
$ TRACEROUTE :== $TCPWARE:TRACEROUTE
```

To trace the path an IP packet follows to an internet host, enter the following format:

```
$ TRACEROUTE [-nrv] [-w wait] [-m max_ttl] [-p port#] [-q nqueries]-
_$ [-t tos] [-s src_addr] host [data size]
```

-n	Specifies to display internet addresses, not host names
-r	Specifies not to route (TRACEROUTE displays only directly "connected" hosts)
-v	Specifies verbose mode
-w ***wait***	Is the number of seconds to wait between probes
-m ***max_ttl***	Is the maximum time-to-live to set in or outgoing datagrams
-p ***port***	Is the UDP port number to use (the default is 33434)
-q ***nqueries***	Is the number of probes sent
-t ***tos***	Is the type of service (the default is 0)
-s ***src_addr***	Is the source internet address to use (the default depends on the interface needed to reach the destination)
host	Is the name of the internet host whose path you want to trace
data_size	Is the size of the UDP packet, which is the amount of data in the packet

Example 28-5 shows a TRACEROUTE command and the resulting output.

Example 28-5 TRACEROUTE Example

```
$ TRACEROUTE nic.ddn.mil

TRACEROUTE to nic.ddn.mil (192.112.36.5), 30 hops max, 38 byte packets
 1  delta.nene.com (192.168.95.126)  0 ms  0 ms  0 ms
 2  process-gw.nene.com (192.168.138.1)  0 ms  10 ms  0 ms
 3  waltham-cr1.bbnplanet.net (131.192.148.49)  20 ms  0 ms  0 ms
 4  cambridge2-cr3.bbnplanet.net (131.192.27.2)  10 ms  10 ms  10 ms
```

```
 5  cambridge2-br1.bbnplanet.net (199.92.129.6)  10 ms  10 ms  10 ms
 6  cambridge1-br2.bbnplanet.net (4.0.2.25)  10 ms  20 ms  10 ms
 7  cambridge1-br1.bbnplanet.net (4.0.1.201)  210 ms  190 ms  210 ms
 8  nyc1-br2.bbnplanet.net (4.0.1.121)  20 ms  30 ms  20 ms
 9  nyc2-br2.bbnplanet.net (4.0.1.154)  30 ms  90 ms  120 ms
10  nyc2-br2.bbnplanet.net (4.0.1.154)  110 ms  40 ms  20 ms
11  nynap.bbnplanet.net (4.0.1.26)  20 ms  40 ms  20 ms
12  niprnet.sprintnap.net (192.157.69.45)  420 ms  90 ms  90 ms
13  137.209.12.1 (137.209.12.1)  90 ms  90 ms  90 ms
14  * 198.26.119.26 (198.26.119.26)  90 ms  100 ms
15  nic.mil (192.112.36.5)  90 ms
```

PART IX Appendixes

Appendix A

NFS-to-OpenVMS Filename Mapping

This appendix is a supplement to the NFS-OpenVMS Client and NFS-OpenVMS Server chapters in this guide.

The following filename mapping rules are necessary because:

- TCPware Client must map (translate) special characters in NFS server filenames that are not valid in OpenVMS filenames.
- TCPware Server must map special characters in filenames users create on the NFS system client host that are not valid in OpenVMS filenames.

OpenVMS disk filenames can be 39 characters long (as can file extensions) and include only the following characters:

`0` through **`9`**, **`A`** through **`Z`**, **`a`** through **`z`**, dollar sign (**`$`**), hyphen (**`-`**), and underscore (**`_`**)

OpenVMS files also include version numbers (after the semicolon) that cannot exceed the value 32767. The Server preserves these version numbers, and hard-links the highest numbered version to an unversioned filename (see the *Note* at the bottom of Table A-2).

NFS filenames can have any of the ASCII characters except the null character (octal 000) and the slash (/, or octal 057), which delimits directory levels. BSD UNIX filenames can have up to 255 characters.

The Client and the Server use the same filename mapping schemes. Three types of mapping schemes are available:

- SRI International mapping, the default scheme between UNIX and OpenVMS systems

Table A-1 shows the default SRI International mapping.

- PATHWORKS non-case-sensitive mapping (NFS-OpenVMS Server only)
- PATHWORKS case-sensitive mapping (NFS-OpenVMS Server only)

Table A-1 NFS-to-OpenVMS Character Sequence Mapping

ASCII character...	Is mapped in OpenVMS to...	With octal value...
Ctrl/A (soh)	$4A	001
Ctrl/B (stx)	$4B	002
Ctrl/C (etx)	$4C	003
Ctrl/D (eot)	$4D	004
Ctrl/E (enq)	$4E	005
Ctrl/F (ack)	$4F	006
Ctrl/G (bel)	$4G	007
Ctrl/H (bs)	$4H	010
Ctrl/I (ht)	$4I	011
Ctrl/J (nl)	$4J	012
Ctrl/K (vt)	$4K	013
Ctrl/L (np)	$4L	014
Ctrl/M (cr)	$4M	015
Ctrl/N (so)	$4N	016
Ctrl/O (si)	$4O	017
Ctrl/P (dle)	$4P	020
Ctrl/Q (dc1)	$4Q	021
Ctrl/R (dc2)	$4R	022
Ctrl/S (dc3)	$4S	023
Ctrl/T (dc4)	$4T	024
Ctrl/U (nak)	$4U	025
Ctrl/V (syn)	$4V	026
Ctrl/W (etb)	$4W	027
Ctrl/X (can)	$4X	030

Table A-1 NFS-to-OpenVMS Character Sequence Mapping (Continued)

ASCII character...	Is mapped in OpenVMS to...	With octal value...
Ctrl/Y (em)	$4Y	031
Ctrl/Z (sub)	$4Z	032
Ctrl/[(esc)	$6B	033
Ctrl/(fs)	$6C	034
Ctrl/] (gs)	$6D	035
Ctrl/^ (rs)	$6E	036
Ctrl/_ (us)	$6F	037
SPACE (sp)	$7A	040
!	$5A	041
"	$5B	042
#	$5C	043
$	$$ (See Rule 9 in Table A-2)	044
%	$5E	045
&	$5F	046
’	$5G	047
(	$5H	050
)	$5I	051
*	$5J	052
+	$5K	053
,	$5L	054
-	same	055
.	. or $5N (See Table A-2)	056
/	not mapped (directory delimiter)	057
0 to 9	same	060 to 071
:	$5Z	072

Table A-1 NFS-to-OpenVMS Character Sequence Mapping (Continued)

ASCII character...	**Is mapped in OpenVMS to...**	**With octal value...**
;	$7B	073
<	$7C	074
=	$7D	075
>	$7E	076
?	$7F	077
@	$8A	100
A to Z	same	101 to 132
[	$8B	133
	$8C	134
]	$8D	135
^	$8E	136
_	same	137
‘	$9A	140
a to z	same	141 to 172
{	$9B	173
|	$9C	174
}	$9D	175
~	$9E	176
DEL	$9F	177
octal 200 to ¿	$200 to $277	200 to 277 (Multinational)
À to octal 377	$300 to $377	300 to 377 (Multinational)

The NFS-to-OpenVMS filename translation rules in Table A-2 are based on the character mapping scheme in Table A-1. The OpenVMS-to-NFS mapping rules are the converse of these rules.

Table A-2 NFS-to-OpenVMS Filename Translation Rules

Rule	What happens to filenames from NFS to OpenVMS...
1	Lowercase characters become uppercase (unless Rule 2 applies; see also Rule 3): **`foobar.txt`** becomes **FOOBAR.TXT;1**
2	Initial uppercase characters or a sequence of case-shifted characters get a `$` prefix: **`CaseShiftedFile`** becomes **CASESHIFTEDFILE.;1**
3	An unversioned file gets a version number preceded by a semicolon: **`foobar.txt`** becomes **FOOBAR.TXT;1**
4	If a filename does not include a file extension dot (.), it gets one before the version number semicolon: **`foobar`** becomes **FOOBAR.;1**
5	After being translated, if a filename (or its extension after the dot) has more than 39 characters or the version number (after the semicolon) is greater than 32767, the file will not show up in an OpenVMS listing.
6	The first dot in a filename is preserved, unless the result fails the 39-character extension limit test in Rule 5 (if so, the dot becomes `$5N`). Each successive dot becomes `$5N`, unless the filename exceeds the limits in Rule 5. **`more.file.txt`** becomes **MORE.FILE$5NTXT;1**
7	If the filename is a directory name, each dot in it becomes `$5N` and the filename gets the `.DIR` extension: **`dot.directory.list`** becomes **DOT$5NDIRECTORY$5NLIST.DIR;1**
8	Invalid OpenVMS characters become the escape character sequences in the second column of Table A-1 (`$` followed by a digit and a letter): **`special#character&file`** becomes **SPECIAL$5CCHAR$5FFILE.;1** (# becomes `$5C` and & becomes $5F)
9	Any existing `$` becomes `$$` (plus any `$` added due to Rule 2 or 8 above): **`dollar$Sign$5cfile`** becomes **DOLLAR$$$S$IGN$$5CFILE.;1**

Note! Many UNIX applications use only unversioned filenames and simply overwrite existing files. Normally, if you create a file with a new version number in OpenVMS, the Client and Server include the version number when displayed on the NFS server. Also, if `foobar;9` is the highest-numbered version in NFS, the Client creates an unversioned `foobar` file and hard-links to it. You can limit this versioning by specifying the /NOVERSION qualifier with the Client NFSMOUNT command; this limits the number of versions created to one, unless an accompanying Attributes Data File (ADF) specifies otherwise.

Keep in mind that the highest numbered version is not necessarily the most recent one, such as when you create a file explicitly with a lower version number than an existing one. For example, the Client and Server normally do not append a version number at the end of a `foobar;1` file and just keep it as `foobar`, unless you explicitly specify `foobar;1` with an already existing higher version of the file.

Appendix B

Data Network Identification Codes

This appendix provides information about known Data Network Identification Codes (DNICs). This appendix is for the network manager.

Each packet switched network has a Data Network Identification Code (DNIC). This code distinguishes the networks from one another.

Table B-1 lists the DNIC range by general location.

Table B-1 DNIC Range by Location

DNIC range...	Has a general location of...
2000 – 2999	Europe
3000 – 3999	North America and Caribbean islands
4000 – 4399	Middle East
4400 – 4999	Asia
5000 – 5999	Philippines, South East Asia, and Pacific islands
6000 – 6999	Africa
7000 – 7499	South America

Table B-2 lists known DNICs by numerical order, the network name, the carrier, the country of origin and its abbreviation, and the rules for communicating within and outside the carrier network. Use this information when you create address and translate entries in the X25.CONF file described in the *X.25 Interface* chapter of this guide.

In Table B-2, the last column for each DNIC has two entries:

Within Network, Remove	Portion of the DNIC to remove if communication is to stay within the local carrier network

Outside Network, Add	Code to add to the full X.25 address if communication is to go outside the local carrier network

If there is no entry for a DNIC, contact your carrier, or consult a Web site such as `www.itu.org.br` (with a search on `dnic`) to receive an updated list.

Use the information from Table B-2 to build the translate portion of X25.CONF file. For example, the translate entry for a DNIC of 3144 would be:

`translate X.25 3144* NYNEX *` (remove 3144 from string for communication within the network)

`translate X.25 * NYNEX 1*` (add 1 to string for communication outside the network)

Make sure you enter the local carrier network name as configured in VAX P.S.I. In this example VAX P.S.I. was configured using the name NYNEX.

Table B-2 DNIC and Carrier Information for Translate Entries

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
2022	HELPAK	Hellenic Telecom Org.	GR	Greece		
2041	DATANET		NL	Netherlands	204	0
2042	EURONET		NL	Netherlands		
2044	DABAS	PTT	NL	Netherlands		
2062	DCS	RTT	BE	Belgium	206	0
2063	EURONET		BE	Belgium		
2064	DCS	RTT	BE	Belgium	206	0
2080	TRANSPAC	PTT	FR	France	2080	
2081	NTI	PTT	FR	France		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
2083	EURONET	PTT	FR	France		
2160	PCTO	PTT	HU	Hungary		
2141	IBERPAC	CTNE	ES	Spain		
2145	IBERBAC	CTNE	ES	Spain	214	
2220		Italcable	IT	Italy		
2222	DARDO	Italcable	IT	Italy		
2222	ITAPAC		IT	Italy	222	
2223	EURONET		IT	Italy		
2284	TELEPAC	Suisse PTT	CH	Switzerland	228	0
2289	DATALINK	Radio Suisse	CH	Switzerland		
2320		Radio Austria	AT	Austria		
2322	DATEX-P		AT	Austria	232	
2329	RADAUS	Radio Austria	AT	Austria		
2341	IPSS	BTI	GB	United Kingdom		
2342	PSS	BT	GB	United Kingdom	234	

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
2343	EURONET		GB	United Kingdom		
2350	MERCURY	Mercury Comm. Ltd.	GB	United Kingdom		
2381	Datex		DK	Denmark		
2382	DATAPAK	PTT	DK	Denmark	238	
2401	TELEPAK	government	SE	Sweden		
2402	DATAPAK	Swedish Telecom	SE	Sweden	2402	
2403			SE	Sweden		
2405	TELEPAK	government	SE	Sweden		
2421	Datex	PTT	NO	Norway		
2422	DATAPAK	PTT	NO	Norway	242	
2441	Datex		FI	Finland		
2442	DATAPAK	PTT	FI	Finland	244	0
2443	Digipak		FI	Finland	None	None
2623	EURONET		DE	Germany		
2624	DATEX-P	DBP	DE	Germany	262	0

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
2680	TELEPAK		PT	Portugal	268	
2682	SAABD	CPMR Telcom Int'l	PT	Portugal		
2703	EURONET		LU	Luxembourg		
2704	LUXPAC	PTT	LU	Luxembourg	270	0
2721	IPSS		IE	Ireland		
2723	EURONET		IE	Ireland		
2724	EIRPAC	Telcom Eireann	IE	Ireland	272	
2740	ICEP	PTT	IS	Iceland		
3020	Datapac	TCTS	CA	Canada	3020	1
3024	GLOBDAT	Teleglobe Ca.	CA	Canada		
3025	GLOBDAT	Teleglobe Ca.	CA	Canada		
3028	INFOSWITCH		CA	Canada	308	
3029	INFOSWITCH	CNCP	CA	Canada		
3101	WUT	Western Union Digital Data	US	United States	3101	0
3102	WUI	WUI Digital Datel	US	United States		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
3103	UDTS II	ITT	US	United States	310	
3104	WUI	WUI Database Access	US	United States		
3104	IDAR		TH	Thailand		
3104	Impac		US	United States	3104	
3105	WUI	WUI Leased Channel	US	United States		
3106	TYMNET	Tymenet	US	United States	3106	
3107	UDTS I	ITT Datel	US	United States	310	
3108	ITT	ITT Short Term Voice/Data	US	United States		
3108	ITT	ITT Short Term Voice/Data	US	United States		
3109	DATEL I	RCA	US	United States		
3110	Telenet	Telenet Communications	US	United States	None	None
3111	DATEL II	RCA	US	United States		
3112	WUT	Western Union Broadband	US	United States		
3113	LSDS	RCA	US	United States		
3114	INFOMASTER	Western Union	US	United States		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
3115	GRAPHNET	Graphnet Interactive	US	United States		
3116	GRAPHNET	Graphnet Store & Forward	US	United States		
3117	WUI Telex	WUI	US	United States		
3118	GRAPHNET	Graphnet Facsimile	US	United States		
3119	TRT	TRT Packet Switching	US	United States		
3120	ITT	ITT Low Speed	US	United States		
3121	FTCC	FTCC Datel	US	United States		
3122	FTCC	FTCC Telex	US	United States		
3123	FTCC	FTCC Leased Channel	US	United States		
3124	FTCC	FTCC Packet Switching	US	United States		
3125	UNINET	ITT Uninet	US	United States		
3126	Autonet	ADP Network Services	US	United States	312	
3127	GTE Telenet	GTE Telenet	US	United States		
3128	TRT	TRT Tekex	US	United States		
3129	TRT	TRT Leased Channel	US	United States		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
3130	TRT	TRT Digital Data	US	United States		
3131	RCA	RCAG Telex	US	United States		
3132	CompuServe	CompuServe	US	United States	313	
3134	ACCUNET	AT&T	US	United States	313	
3135	ALASKANET	Alascom	US	United States		
3136	Marknet		US	United States	3136	
3137	Infonet	Computer Sciences Corporation	US	United States	3137	
3140	ConnNet	Southern New England Telephone	US	United States	314	
3141	Bellatlantic		US	United States	3141	
3142	Pulselink		US	United States	314	
3144	Infopath	NYNEX	US	United States	3144	1
3300	UDTS	ITT	PR	Puerto Rico, U.S.A.		
3300	RCA/PR	RCA	PR	Puerto Rico, U.S.A.		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
3301	PRTC	Puerto Rico TelCo	PR	Puerto Rico, U.S.A.		
3320	UDTS	ITT	VI	Virgin Is., U.S.A.		
3340	TELEPAC	PTT	MX	Mexico	334	
3380	Jamantel	Jamaica Int'l Telecom Ltd.	JM	Jamaica		
3400	TRANSPAC	French PTT	GP	Fr. Antilles		
3400	DOMPAC	Ag. Comm. des Telecom	GP	Fr. Guadeloupe		
3400	DOMPAC	Ag. Comm. des Telecom	MQ	Fr. Martinique		
3423	IDAS	Barbados Ext'l Telecom Ltd.	BB	Barbados		
3443	AGANET	C&W (West Indies)	AN	Antigua		
3463	C&W CAYMAN	C&W CAYMEN	KY	Cayman Islands		
3503	C&W	C&W (West Indies)	BM	Bermuda		
3640	BATELCO	Bahamas Telecom Corp.	BS	Bahamas		
3700	UDTS	ITT	DO	Dominican Republic		
3740	TEXTEL	T&T External	TT	Trinidad & Tobago		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
4180	IDAS	BTC	IQ	Iraq		
4200	IDAS	BTC	SA	Saudi Arabia		
4270	IDAS	BTC	QA	Qatar		
4251	ISRANET	PTT	IL	Israel		
4263	ODAS (BAHNET)	Bahrain Telecom Co. (BTC)	BH	Bahrain		
4310	TEDAS	Emirates Telcom Corp.	AE	United Arab Emirates		
4401	DDX-P	NTT	JP	Japan	4401	1
4406	NIS/ TYMNET	Network Information Service	JP	Japan	4406	
4408	VENUS-P	KDD	JP	Japan	440	
4501	DACOMNET	Data Comm. Corp	KR	Korea, S. (ROK)	450	
4503	DACOM		KR	Korea, S. (ROK)		
4542	IDAS/ITS	C&W	HK	Hong Kong		
4542	INTELPAK	C&W	HK	Hong Kong	454	
4544	DAS	C&W	HK	Hong Kong		

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
4545	DATAPAK		HK	Hong Kong	454	
4600	PKTELCOM	Beijing Telecom Adm.	CH	China		
4872	Pacnet		TW	Taiwan	487	
4877	UDAS	ITA	TW	Taiwan		
5021	MAYPAC		MA	Malaysia	502	
5052	AUSTPAC	Telecom Australia	AU	Australia	505	
5053	MIDAS	OTC	AU	Australia		
5101	SKDP	PTT	ID	Indonesia	510	0
5150	ETPI	Eastern Telecom	PH	Philippines		
5150	UDTS	Globe Macay C&R	PH	Philippines		
5150	PCG	Phil. Global Comm., Inc.	PH	Philippines		
5200	CAT	CAT	TH	Thailand		
5252	TELEPAC	Telecoms Singapore	SG	Singapore	525	0
5301	IPSS	PTT	NZ	New Zealand	530	
5301	PACNET		NZ	New Zealand	530	

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
5350	RCA	RCA	GU	Guam, U.S.A.		
6020	ARENTO	Telecom Org. of Egypt	EG	Egypt		
6122	SYTRANPAK	Intelci	CI	Ivory Coast		
6282	GABONPAC	Telecom Int’l	GA	Gabon		
6470	DOMPAC	Ag. Comm. des Telecom	RE	Fr. Reunion		
6550	SAPONET	SAPO	ZA	South Africa		
7080	HONDUTEK	Empresa Hondurena de Telecom	HN	Honduras		
7120	RACSAPAC	Radiográfica Costarricense	CR	Costa Rica		
7141	INTEL	Instituto Nac. de Telecom	PA	Panama		
7160	ENTEL	Empresa Nac. de Telecom	PE	Peru		
7220	ENTEL	Empresa Nac. de Telecom	AR	Argentina		
7222	ARPC		AR	Argentina	722	
7240	INTERDATA	EMBRATEL	BR	Brazil		
7241	RENPAC		BR	Brazil	724	

Table B-2 DNIC and Carrier Information for Translate Entries (Continued)

DNIC	Network Name	Carrier	Country		Within Network, Remove:	Outside Network, Add:
7320	DATAPAK	Empresa Nac. de Telecom	CO	Colombia		
7420			GF	Fr. Guyana		

INDEX

Symbols

A

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Reader's Comments
TCPware for OpenVMS Version 5.4 Management Guide, Part Number: N-5200-02-NN-A

Your comments and suggestions will help us to improve the quality of our future documentation. Please note that this form is for comments on documentation only.

I rate this guide's:	**Excellent**	**Good**	**Fair**	**Poor**
Accuracy	o	o	o	o
Completeness (enough information)	o	o	o	o
Clarity (easy to understand)	o	o	o	o
Organization (structure of subject matter)	o	o	o	o
Figures (useful)	o	o	o	o
Index (ability to find topic)	o	o	o	o
Ease of use	o	o	o	o

1. I would like to see more/less:____

2. Does this guide provide the information you need to perform daily tasks?________________________________

3. What I like best about this guide:____

4. What I like least about this guide:____

5. Do you like this guide's binding? If not, what would you prefer?_____________________________________

My additional comments or suggestions for improving this guide:____
____

I found the following errors in this guide:

Page	**Description**
________	____
________	____

Please indicate the type of user/reader that you most nearly represent:

System Manager	o	Educator/Trainer	o
Experienced Programmer	o	Sales	o
Novice Programmer	o	Scientist/Engineer	o
Computer Operator	o	Software Support	o
Administrative Support	o	Other (please specify)	o ____________________________

Name:____ Dept.___________________________
Company:____ Date___________________________
Mailing Address:____

After filling out this form, FAX or mail it to:

Process Software Corporation, 959 Concord Street, Framingham, MA 01701-4682
Attention: Technical Publications Group FAX 508-879-0042 e-mail:techpubs@process.com