TCPware 6.1 User's Guide

September 2023

This manual describes how to use the network services provided by TCPware.

Operating System/Version: OpenVMS VAX V5.5-2 or later
OpenVMS Alpha V6.2 or later
OpenVMS Itanium V8.2 or later

Software Version: TCPware 6.1

Process Software
Framingham, Massachusetts
USA

Preface

Introducing This Guide

This guide describes the TCPware products, components, and features, and the user environment and functions. It is an introduction for all users, as well as a procedural guide for end users.

What You Need to Know Beforehand

Before using TCPware, you should be familiar with:

- Computer networks in general

- The OpenVMS operating system and file system

How This Guide Is Organized

This guide has the following contents:

- Part I, *Introduction* - Introduces and provides a functional overview of the TCPware product, components, and features.
- Part II, *User Functions* - Provides user instructions for the following TCPware components and features, arranged in chapters alphabetically:
  - FTP
  - Kerberos authentication user commands
  - Network print functions (Line Printer Services and Terminal Server Print Services)
  - Remote Compact Disk (RCD) and Remote Magnetic Tape (RMT)
  - Remote Copy Program (RCP)
  - RLOGIN
  - Remote Shell (RSH)
  - Simple Mail Transfer Protocol (SMTP)
  - TALK
  - TELNET
  - Trivial File Transfer Protocol (TFTP)
  - Token Authentication User Functions
  - WHOIS
  - Secure Shell (SSH)
- Appendices, including a list of references and a glossary of terms.

Online Help

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

- Topical help - Access TCPware help topics 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 client and server, Network Control Utility (NETCU), TELNET client, NSLOOKUP, and TRACEROUTE. Use the `HELP` command from within each component:

```
NETCU>HELP [topic]
```

- Error messages help - Access help for TCPware error messages 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
- Process Software support contract 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 Technical Support toll-free at (800) 394-8700. If calling from outside the continental United States or Canada, dial +1 (508) 628-5074. Please be ready to provide your name, company name, Process Software support contract number, and telephone number.

World Wide Web

There is a variety of useful technical information available on our World Wide Web home page, http://www.process.com/

Documentation Set

The documentation set for TCPware consists of the following:

- ***Installation & Configuration Guide*** - For system managers and those installing the software. The guide provides installation and configuration instructions for the TCPware products.
- ***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 and application programs.
- ***Release Notes*** for the current version of TCPware - For all users, system managers, and application programmers. The *Release Notes* are available online on your TCPware media and are accessible before or after software installation.
- ***User's Guide*** - For all users. This guide includes an introduction to TCPware products as well as a reference for the user functions arranged alphabetically by product, utility, or service.

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 **`reversed 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.
`key`	Press the specified key on your keyboard.
`Ctrl+key`	Press the control key and the other specified key simultaneously.
`Return`	Press the Return or Enter key on your keyboard.

1. Introducing TCPware for OpenVMS

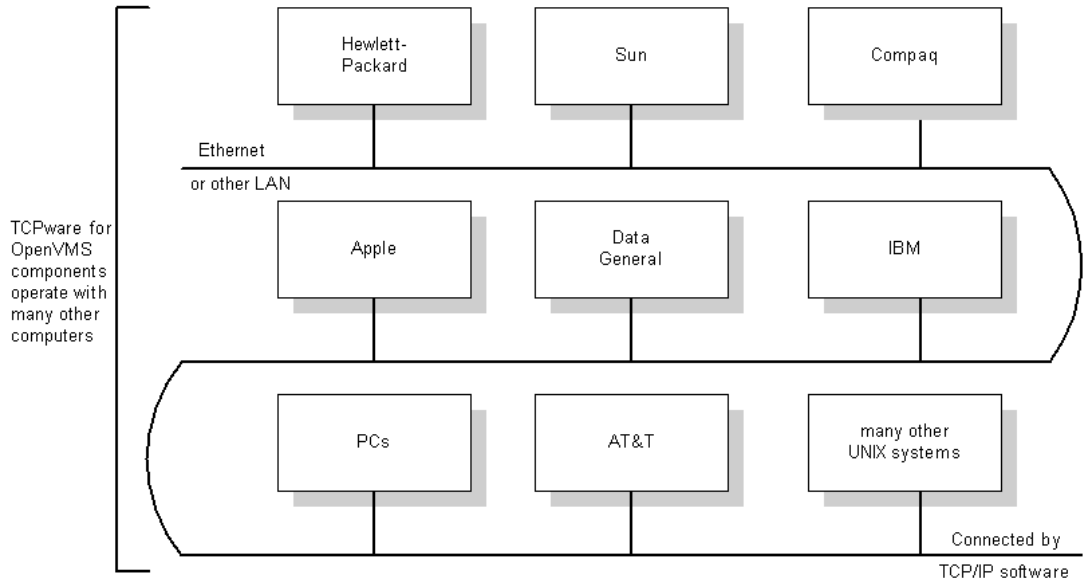
Introduction

TCPware for OpenVMS is a software product that provides TCP/IP standard networking services for HP's OpenVMS VAX, Alpha and Itanium computers.

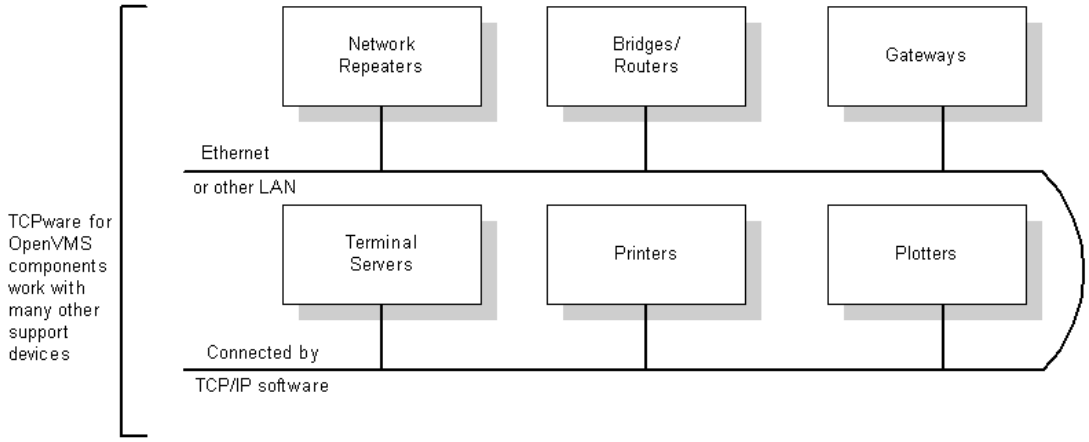
Enterprise-Wide Networking

Computer systems from many different vendors can communicate with systems using the TCP/IP protocols. Almost all UNIX-based systems support TCP/IP, FTP, NFS, SMTP, and TELNET. This makes TCPware for OpenVMS components ideal tools for networking OpenVMS systems with other computer systems.

The below figure shows some systems networked using TCP/IP.



TCPware components operate with many other computers. TCPware components also operate with many network support devices that are compatible with TCP/IP, Ethernet, and other local area networks (LANs), as shown in the below figure.



TCPware for OpenVMS

TCPware includes services components designed exclusively for the VAX, Alpha and Itanium architectures and the OpenVMS operating system.

The below table lists the members of the TCPware family and the features of each.

Component	Features
FTP	File transfer service that lets you transfer files to or from remote hosts. Provides a File Transfer Protocol (FTP) client and server. Includes the Remote Copy Program (RCP) (which includes optional Kerberos authentication). Also includes a subroutine library to develop FTP application programs. Token authentication is also available for FTP.
NFS Client	Network File System (NFS) service that lets you access NFS filesystems and store data on NFS systems. Provides an NFS client.
NFS Server	NFS service that lets remote NFS users access OpenVMS filesystems and use them for storage. Provides an NFS server.
SMTP	Mail transfer service that lets you send mail to or receive mail from remote hosts. Provides a Simple Mail Transfer Protocol (SMTP) client and server. The additional Internet Message Access Protocol (IMAP) and Post Office Protocol Version 3 (POP3) servers provide a way for remote PCs to retrieve OpenVMS incoming mail.

SSH	Secure Shell provides encrypted remote access to this system and other systems with SSH software. Commands may be executed remotely or remote interactive sessions may be used. Files may be transferred with the SCP command, which uses SSH for access to the remote system.
TELNET	Virtual terminal service that lets you have immediate access to remote systems. Provides a Virtual Terminal Networking (TELNET) protocol client and server. Kerberos authentication is also available. Also includes a subroutine library to develop TELNET application programs. Token authentication is also available for TELNET.
TCP	TCP/IP base component that includes protocols for the network layer (IP, ICMP, ARP, and RARP) and transport layer (TCP and UDP). Provides utilities for network management and control: • For Domain Name Services (DNS), Simple Network Management Protocol (SNMP) Services, Network Control Utility (NETCU), and Network Time Synchronization, see the *Network Management* entry in *TCP/IP Services***Error! Reference source not found.**. • Berkeley R Commands - Access hosts in a TCP/IP network by logging in (RLOGIN), executing remote commands (RSH), and controlling remote tape drives (RMT) and CD-ROM drives (RCD). Token authentication is also available for RLOGIN. • Line Printer Services - Manipulate local or remote print queue functions based on the client and server ends of the Line Printer Protocol. • Terminal Server Print Services - Send print requests to printers attached to TCP/IP-based terminal servers. • Subroutine Libraries - Facilitate application development using the Socket Library Services, FTP subroutine library, TELNET Subroutine Library, and SNMP Extendible Agent Application Program Interface (API) routines. • TCPDRIVER, UDPDRIVER, IPDRIVER, and INETDRIVER programming services, and UCX compatibility services (BGDRIVER) - Use QIO interfaces to develop network applications. UCX compatibility allows applications such as PATHWORKS to work with TCPware. • ONC RPC Services - Build distributed applications using Remote Procedure Calls (RPCs).

TCP/IP Services

All of TCPware's TCP/IP services are fully integrated. The services range from the upper-layer Network Application Services to the lower-level components. These lower-level components handle the network controllers included in the TCP/IP services core component, TCP-OpenVMS.

The TCPware components use the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Protocol (IP). TCP and IP provide a reliable and efficient means for moving information between computer systems.

TCPware supports Path MTU discovery to provide a performance improvement when large packets of data are sent over TCP.

The below table describes some of the TCP/IP functions supported by TCPware.

This Service...	**Provides...**
Cluster Load Balancing	Have the domain name server assign a connection to a specific host to balance the cluster load. Analogous to the load balancing services the LAT terminal service provides.
Database Support	Connect Ingres, Oracle, RDB, Progress, and Sybase databases on OpenVMS and UNIX systems.
DECnet over IP	Send DECnet data link layer packets point-to-point over TCP/IP connection between two systems running TCPware.
DECwindows	Supports DECwindows graphics-oriented applications like Mail, File View, DECterm, and Bookreader. A remote X display user can also log in using the X Display Manager Server.
Interface Support	Interface support, which includes: • **Ethernet, Token Ring, and LAT interfaces** - Send IP datagrams over Ethernet, Token Ring, LAN Emulation over Asynchronous Transfer Mode (ATM), and Classical IP over ATM (CLIP) networks. Supports the Address Resolution Protocol (ARP) and Reverse ARP (RARP). • **Fiber Distributed Data Interface (FDDI)** - Send IP datagrams over high-speed networks over FDDI controllers. Supports ARP and RARP.

	• **HYPERchannel** - Directly supports the UNIBUS, QBUS, MASSBUS, and BIBUS interfaces. Includes use of ARP to map host internet addresses to physical addresses. • **IP-over-DECnet** - Send IP datagrams over DECnet links to connect separate DECnet-over-IP TCP/IP LANs over WANs. • **IP-over-X.25** - Send IP datagrams as data packets over X.25, enabling reliable worldwide communication. • **Point-to-Point Protocol (PPP)** - Send multiprotocol datagrams over serial point-to-point links. PPP is common with line speeds from 14.4 to 28.8 kilobits per second (Kbps). Implemented through `pppd` command line options. • **proNET** - Supports the proNET-10 and proNET-80 token ring controllers provided by Proteon, Inc. • **Serial Line IP (SLIP)** - Send IP datagrams over serial lines instead of Ethernet cable. Supports both dedicated (hard-wired) and dialup SLIP lines. TCPware also supports Compressed SLIP (CSLIP). • **HP Wide Area Network (WAN) Device Drivers** - Supports the WAN Device Drivers synchronous interfaces that form a link between the hardware devices and TCPware.
Multicasting	Supports full IP multicasting, letting you send and receive datagrams addressed to IP multicast (Class D) addresses. Implements the Internet Group Management Protocol (IGMP).
Network Management	Network management and control functions include: • **Domain Name Services (DNS)** - Guarantees host connections using a distributed database. • **Dynamic Host Configuration Protocol (DHCP)** - Provides IP addresses and configuration data to hosts. Supports DHCP and BOOTP protocols. • **Simple Network Management Protocol (SNMP) Services** - Network management stations can obtain timely information about the network activities of OpenVMS server hosts. Supports MIB-I and MIB-II. TCPware's SNMP agent also supports subagents serving private MIBs, as well as the SNMP Multiplexing (SMUX) Service. • **Network Control Process (NETCP)** - Starts, maintains, and shuts down the network. NETCP also contains the Port Mapper that maps Remote Procedure Call (RPC) server programs to ports. A TCPDUMP utility is also included. • **Network Control Utility** (NETCU) - Provides commands so that the system manager can monitor and control various functions such as adding and removing servers and clients.

	• **Network Time Synchronization** - Use either the Network Time Protocol (NTP) or the Time Synchronization Protocol (TIMED), to coordinate time distribution between hosts.
Network Security	Includes Incoming and Outgoing Access Restrictions; Packet Filtering; user commands, management commands, and administration server; the IP Security Option (IPSO); and Token Authentication for login security.
Other Clients and Servers	Client protocols (DISCARD, FINGER, NSLOOKUP, PING, TALK, Trivial File Transfer Protocol [TFTP], TRACEROUTE, and WHOIS) and Server protocols (CHARGEND, DAYTIMED, DISCARDED, ECHOD, FINGERD, INDENT, QUOTED, and TFTPD).
PATHWORKS Support	Use TCPware as a transport for HPE's PATHWORKS products running between the OpenVMS system and a PC.
Routing	Supports enhanced routing and multiple gateways, and includes the GateD protocol, which combines RIP, HELLO, OSPF, EGP, BGP, and the Router Discovery Protocol for distributing routing information. Supports the Classless Inter-Domain Routing (CIDR) protocol for more efficient use of IP addresses.

2. Functional Overview

Introduction

This chapter presents a functional overview of the TCPware components. It addresses questions you may have, such as what you use to:

- Access to network filesystems as if they were local filesystems
- Transfer (copy) files over the network
- Print network files
- Log in to and perform commands on a remote system
- Send or receive mail or message over the network
- Access to network magnetic tape or CD-ROM drives
- Dynamically configure network hosts and find network information

- Control network activity
- Synchronize clocks across the network
- Secure resources on the network
- Tunnel external protocol applications over IP
- Program network interfaces

For more details on each subject, we provide you with references to the appropriate section of this documentation set at the end of this chapter.

Remote Filesystem Access

You can access remote filesystems as if they were your own, using NFS.

This component...	Allows you to...	To use it, you need...	As a user...	As a system manager...
NFS client	On a TCP/IP network, transparently access filesystems on remote servers so that they appear as resident filesystems in OpenVMS.	To access remote filesystems, run the NFS client. You must have authorization to access them.	simply use the filesystems as if they were on your local system. No special commands are required.	see the *Management Guide*, Chapter 13, *Managing NFS Client*.
NFS server	Provide a service so that remote system users can access your local OpenVMS filesystems as if they were their own.	For remote systems users to access OpenVMS files on your system, run the NFS server. The remote user must have authorization to access your local filesystems.		see the *Management Guide*, Chapter 14, *Managing NFS Server*.

Transferring Files

You can transfer files to or from your OpenVMS system using FTP (which includes the RCP feature) or TFTP.

This component...	Allows you to...	To use it, you need...
FTP	Copy, get, and put files to and from remote systems using the File Transfer Protocol (FTP). TCPware provides both the client function so that local users can transfer files	The remote system must support FTP.

	to and from remote systems, and the server function so that remote users can transfer files from your local system.	As a user, see the *User's Guide*, Chapter 3, *FTP: Transferring Files*. As a system manager, see the *Management Guide*, Chapter 12, *Managing FTP*. As a system programmer, see the *Programmer's Guide*, Chapter 7, *FTP Library*.
RCP	Use a UNIX-like command to copy files to and from remote systems right on the system command line. TCPware also provides the RCP server so that remote users can copy files to or from your system.	The server must support equivalents of the UNIX `shell` and `exec` services. You must register the other hosts in your `HOSTS.EQUIV` or `.RHOSTS` files. As a user, see the *User's Guide*, Chapter 7, *RCP: Copying Files*. As a system manager, see the *Management Guide*, Chapter 16, *Managing R Commands*.
TFTP	Transfer files to and from remote systems. Because TFTP is more primitive than FTP, you can mainly use TFTP to allow remote diskless systems to read bootstrap images over the network.	The remote system must support TFTP. As a user, see the *User's Guide*, Chapter 13, *TFTP: Trivial File Transfers*.

		As a system manager, see the *Management Guide*, Chapter 16, *Managing R Commands*.

Printing Files

You can print files over the network using the Line Printer Services or Terminal Server Print Services.

This component...	**Allows you to...**	**To use it, you need...**	**As a user...**	**As a system manager**...
Line Printer Services	Send files to, remove jobs from, and display the status of remote print queues using UNIX-like commands. Line Printer Services also provides a server so that remote users can access local print queues.	to define the remote printers during installation.	see the *User's Guide*, Chapter 5, *Networking Printing*.	see the *Management Guide*, Chapter 15, *Managing Print Services*.
Terminal Server Print Services	If you are on a TCP/IP network, send files to printers connected to remote terminal servers.	Use the regular `PRINT/QUEUE` commands.	see the *User's Guide*, Chapter 5, *Network Printing*.	see the *Management Guide*, Chapter 15, *Managing Print Services*, the *Terminal Server Print Services* section.

Logging In to Remote Hosts

You can log in to and execute commands on remote hosts using the RLOGIN or RSH features of TCPware.

This component...	Allows you to...	As a system programmer...	As a user...	As a system manager...
RLOGIN	Use a UNIX-like command to log in to a remote host.		see the *User's Guide*, Chapter 8, *RLOGIN: Logging In to a Remote Host*.	see the *Management Guide*, Chapter 16, *Managing R Commands*.
RSH	Use a UNIX-like command to execute a single command on a remote host without logging in.		see the *User's Guide*, Chapter 9, *RSH: Issuing Commands on the Remote Host*.	see the *Management Guide*, Chapter 16, *Managing R Commands*.
TELNET-OpenVMS	Initiate virtual terminal connections to remote hosts using the TELNET protocol. You can open multiple remote sessions. TCPware also provides a server function so that remote users can make virtual terminal connections to the OpenVMS host.	see the *Programmer's Guide*, Chapter 9, *TELNET Library*.	see the *User's Guide*, Chapter 12, *TELNET: Connecting to Remote Terminals*.	see the *Management Guide*, Chapter 18, *Managing TELNET Server*.

Transferring Mail and Exchanging Messages

You can send and receive mail over the network using the TCPware SMTP, IMAP, and POP3 components.

This component...	Allows you to...	To use it, ...	As a user...	As a system manager...
SMTP	On a TCP/IP network, send and receive mail over the network using the Simple Mail Transfer Protocol (SMTP). TCPware provides both an SMTP client and a server.	The remote system must support SMTP.	see the *User's Guide*, Chapter 10, *SMTP: Transferring Mail*.	see the *Management Guide*, Chapter 17, *Managing Mail Services*.
IMAP Server	Provide a service so that remote PCs can access mail in VMS MAIL mailboxes using the Internet Message Access Protocol (IMAP) Server.	The remote system must support the IMAP protocol.		see the *Management Guide*, Chapter 17, *Managing Mail Services*, the *IMAP Server* section.
POP3 Server	Provide a service so that remote PCs can retrieve mail in VMS MAIL in-boxes using the Post Office Protocol (POP3) Server.	The remote system must support the POP3 protocol.		see the *Management Guide*, Chapter 17, *Managing Mail Services*, the *POP3 Server* section.
TALK Utility	Exchange "real time" messages with another host on the local or remote	The remote system must support the	see the *User's Guide*, Chapter 11,	

	network. Display simultaneously sent and received messages on a split screen.	`talk` protocol.	*TALK: Exchanging Terminal Messages*.	

Accessing Network Drives

You can access remote tape or CD-ROM drives, or provide access locally to remote users by using the TCPware RMT and RCD components.

This component...	**Allows you to...**	**To use it, you need to configure...**	**As a user...**	**As a system manager...**
RMT Client	Use OpenVMS commands such as `BACKUP`, `MOUNT`, `COPY`, and `EXCHANGE` on remote backup tape drives.	a pseudo-device on your OpenVMS system using the command `RMTSETUP`. The remote system must support the RMT protocol.	see the *User's Guide*, Chapter 6, *RCD and RMT: Remote CD-ROMs and Tapes*.	
RCD Client	Use OpenVMS commands such as `BACKUP`, `MOUNT`, `COPY`, and `EXCHANGE` on remote CD-ROM drives.	a pseudo-device on your OpenVMS system using the command `RMTSETUP`. The remote system must support the RMT protocol.	see the *User's Guide*, Chapter 6, *RCD and RMT: Remote CD-ROMs and Tapes*.	
RMT Service	Provide a service so that remote clients can use the `rdump` or `rrestore` UNIX	the Berkeley R Commands for RMT services. The remote		see the *Management Guide*, Chapter 16,

	utilities to access a magnetic tape on your system.	system must support the RMT protocol.		*Managing R Commands*.

Configuring Hosts

TCPware provides various components and features with which you can configure network hosts, as listed below.

This component...	Allows you to...	As system manager...
DHCP/BOOTP	Assign IP addresses and provide configuration data to hosts over the network.	see the *Management Guide*, Chapter 2, *DHCP/BOOTP Server*.
Domain Name Services	Obtain information such as host Internet addresses and names by connecting to a distributed database.	see the *Management Guide*, Chapter 3, *Domain Name Services*.
Point-to-Point Protocol (PPP)	Configure the network to send IP datagrams over serial links, including DECnet or modern connections.	See the *Management Guide*, Chapter 5, *Serial Link Interfaces: PPP and SLIP*.
Serial Line IP (SLIP) Protocol	Further configure the network to send IP datagrams over serial links.	

Controlling Network Functions

You can perform network management functions and test networks by using the TCPware features listed below.

This component...	Allows you to...	As a system manager...
Network Control Utility (NETCU)	NETCU is the utility program system managers and user use to configure and control network activity.	see the *NETCU Command Reference*.
Simple Network Management Protocol (SNMP) Services	Obtain timely information about network activities of OpenVMS server hosts, such as routing, line status, volume of traffic, and error conditions. SNMP supports the MIB-I and MIB-II Management Information bases, as well as SNMP Multiplexing (SMUX) and SNMP Agent eXtensibility (AGENTX).	see the *Management Guide*, Chapter 7, *Managing SNMP Services*. see the *Programmer's Guide*, Chapter 10, *SNMP Extendible Agent API Routines*.

Synchronizing Time Clocks

TCPware provides the network time synchronization components listed below.

This component...	On a TCP/IP network, allows you to...	As a system manager, see the *Management Guide*,
Network Time Protocol	synchronize your system clock with an Internet Time Server.	Chapter 10, *Network Time Protocol (NTP).*
TIMED	use the Time Synchronization Protocol (TSP) and the `timed` service to synchronize the clocks of LAN hosts.	Chapter 11, *TIMED*.

Using Network Testing Tools

TCPware provides various network testing tools, and utilities and services with which you can obtain network information, as listed below.

This component...	Allows you to...	As a user...	As a system manager...
FINGER	Extract user information from a remote user information program.		See the *Management Guide*, Chapter 30, *Network Testing Tools*.
IDENT	Determine the user associated with a connection.		See the *Management Guide*, Chapter 30, *Network Testing Tools*.
NSLOOKUP	Extract information about network hosts from the Domain Name Systems.		See the *Management Guide*, Chapter 30, *Network Testing Tools*.
PING	Find out if a host is up and if you can reach it.		See the *Management Guide*, Chapter 30, *Network Testing Tools*.
TCPDUMP Utility	Track TCP packets by printing information in packet headers.		See the *Management Guide*, Chapter 30, *Network Testing Tools*.
TRACEROUTE	Trace the path of an IP packet to an internet host.		See the *Management Guide*, Chapter 30, *Network Testing Tools*.
WHOIS	Query the Network Information Center (NIC) username directory services to obtain usernames.	See the *User's Guide*, Chapter 15, *WHOIS: Username Directory Services*.	

TCPware also provides other useful testing utilities and services, including CHARGEND, DAYTIMED, DISCARD, ECHOD, NETCU DEBUG, QUOTED, and TIME. See the *Management Guide*, Chapter 29, *Network Testing Tools*, for details.

Securing Resources

You can secure resources on the network using the TCPware features described below.

This component...	**Allows you to...**	**As a system manager, see the *Management Guide,***	**As a user, see the *User's Guide,***
Incoming Access Restrictions	Restrict the hosts and networks that can access the services the master server activates.	Chapter 20, *Access Restrictions*.	
Outgoing Access Restrictions	Restrict requests for remote services to specific users and ports.	Chapter 20, *Access Restrictions*.	
Packet Filtering	Restrict the datagrams a network interface can receive by protocol, source and destination address, or destination port. Use convenient NETCU commands.	Chapter 21, *Packet Filtering*.	
IP Security Option (IPSO)	Provide IP datagram protection using the IP Security Option (IPSO) protocol.	Chapter 24, *IP Security Option (IPSO)*.	
Secure Shell (SSH)	Configure and maintain the TCPware Secure Shell (SSH) server. This is the server side of the software that allows secure interactive connections to other	Chapter 25, *Configuring the Secure Shell (SSH) Server*.	Chapter 16, *Accessing Remote Systems with the Secure*

	computers in the manner of rlogin/rshell/telnet.		*Shell (SSH) Utilities*

Tunneling External Applications over IP

You can tunnel DECnet applications over IP networks if you are using DECnet Phase IV. A connection established between two systems running different protocols is known as a tunnel.

This component...	**Allows you to...**	**As a system manager, see the *Management Guide,***
Tunneling DECnet over IP (for DECnet Phase IV)	Connect two DECnet networks over an IP link. Use with DECnet Phase IV only. There is no need to use this feature with DECnet/OSI (DECnet Phase V).	Chapter 27, *Tunneling DECnet over IP*.

Programming Network Interfaces

If you are a network programmer, you can perform programming functions using the programming interfaces discussed in the Programmer's Guide.

This component...	**Allows you to...**	**As a system programmer, see the *Programmer's Guide,***

FTP Library	Use a programming interface to the FTP protocol. Use the FTP library routines in your own applications to provide FTP capabilities.	Chapter 7, *FTP Library*.
Socket Library	Use either the HP C Socket Library (for OpenVMS Version 5.3 and later) or the TCPware Socket Library (for earlier versions or if you are using the Remote Procedure Call routines).	Chapter 8, *Socket Library*.
TELNET Library	Use a programming interface to the TELNET protocol. Use the TELNET library routines in your own applications to provide TELNET capabilities.	Chapter 9, *TELNET Library*.
UCX Compatibility Services	Use the BGDRIVER $QIO programming interface for compatibility with HP's TCP/IP Services (formerly UCX) product.	Chapter 2, *UCX Compatibility Services*.
QIO Programming Interfaces	Use $QIO programming interfaces to TCP/IP. These include the BGDRIVER, TCPDRIVER, UDPDRIVER, IPDRIVER, and INETDRIVER interfaces.	Chapter 10, *SNMP Extendible Agent API Routines*.
SNMP Extendible Agent Application Programming Interface (API) Routines	Use API routines required for an application program to export private Management Information Bases (MIBx) using the TCPware SNMP agent.	Chapter 10, *SNMP Extendible Agent AAPI Routines*.

3. FTP: Transferring Files

Introduction

The File Transfer Protocol (FTP) transfers files to and from a remote host.

The FTP client utility is your interface to the FTP server. You can run the FTP client interactively or through a startup command procedure.

For TCPware's FTP service to operate between two hosts, the remote host must provide a compliant client or server. You can run FTP directly (interactively) or indirectly from a command procedure. The FTP client supports multiline recall of up to 20 lines.

Before Using FTP

Before you can transfer files, you need:

- To make sure that the TCPware FTP software is installed, configured, and started on your system.
- The name or internet address of the remote host to which you want to connect.
- The username and password of the account on the remote host. If the remote host does not support multiuser protection features, you might not need a username and password.
- The file naming conventions on the remote host.

FTP Session

A typical FTP session consists of the following steps:

1. Open the FTP connection.
2. Determine the format of the files you want transferred.
3. Transfer files using the `GET` (`MGET`), `PUT` (`MPUT`), or `COPY`. The default file format is formatted ASCII.
4. Close or exit the FTP connection.

Features

TCPware's FTP implementation includes the following features:

- Command line execution.
- Informational and error status messages.
- Support of wildcards in source file specifications.

The below table describes some of the features of the TCPware FTP client.

This feature...	**Means that...**
Command Line User Interface	The FTP client allows you to execute FTP commands at the `FTP>`. You can use either DCL-style syntax or UNIX-style syntax at the `FTP>` prompt. DCL syntax can include qualifiers: `FTP>`**`DIRECTORY *.DIR /BRIEF`** You usually enter UNIX-style commands in lowercase: `FTP>`**`ls *.dir`**
Status Messages	The FTP client issues informational and error messages. These messages are self-explanatory and conform to the standard OpenVMS message format. The numeric codes that prefix these messages conform to the RFC 959 standard for FTP.
Wildcards	The FTP client supports wildcards for the `COPY`, `GET`, `PUT`, `DELETE`, and `DIRECTORY` commands. The acceptable wildcard characters are: • Percent sign (%) or question mark (**?**) to represent individual characters. • Asterisk (*) to represent multiple characters. If you include the asterisk wildcard to represent multiple files to FTP, use the `MGET`, `MPUT`, or `MDELETE` commands, or specify the `/MULTIPLE` qualifier

	with the GET, PUT, COPY, or DELETE command. These two examples produce identical results: FTP>**MGET *.TXT** FTP>**COPY *.TXT/MULTIPLE/REMOTE *** Note that you do not require the asterisk for the destination with MGET, but you do require it with COPY. If enclosed in a quoted string, wildcard symbols no longer act as wildcards.

Opening a Connection

Only one FTP connection can be open at a time. Once open, all file transfers and other remote operations use that connection.

You can open an FTP connection by using the command line user interface. Enter one of the following at the DCL prompt:

```
$ FTP
FTP>OPEN host
```

host is the name of the host to which you want to connect. Respond to the login prompts, if any, of the remote host. After a successful login, the FTP> prompt appears where you enter the FTP commands described in the following sections. This is the option shown in the below example.

```
$ FTP[/TLS] host
FTP>
```

host is the name of the host to which you want to connect. Respond to the login prompts, if any, of the remote host. After a successful login, the FTP> prompt appears where you enter the FTP commands described in the following sections. If /TLS is included on the command line, then TLS authentication will be used before user authentication is entered.

```
$ FTP[/TLS] host username password
FTP>
```

Enter the host to which you want to connect, the username of the account on the remote host, and the password of the account on the remote host as part of the command. After a successful login, the FTP> prompt appears where you enter the FTP commands described in the following sections.

```
(Eta) $ FTP                    [1]
FTP> OPEN THETA
_Username: SMITH
_Password:
FTP> GET TEST.TXT
.
.                              [2]
.
FTP> CLOSE
```

Closing and Exiting

An FTP connection remains open until you quit or exit FTP, close the connection, or open a new connection. To close an FTP connection, use one of the following commands:

1. Closes the current connection and continues the FTP session for the next command.

```
FTP>CLOSE
```

2. Both `OPEN` and `CONNECT` close the current connection and open another one.

```
FTP>OPEN host
FTP>CONNECT host
```

3. Exits an FTP session:

```
FTP>EXIT
```

See the `CLOSE`, `OPEN`, and `EXIT` commands in the *Command Reference*.

Examples of closing an FTP connection:

```
(Eta) $ FTP
FTP> OPEN THETA
_Username [smith]: REMOTE_SMITH
_Password:
FTP>
FTP> GET TEST.TXT      [1]
.
.
.
FTP> CLOSE             [2]
FTP> EXIT
(Eta) $
```

Checking Directories

After you establish an FTP connection, you can check the directories on the remote or local host to locate the file(s) you want.

To check remote directories and determine the file format type when in FTP (see the below example):

1. Open the FTP connection and enter:

```
FTP>DIRECTORY
```

Use the CD or SET DEFAULT /REMOTE command to move to other directories on the remote host.

2. Check file extensions to determine file types. You might need to enter special qualifiers when you transfer certain types of files.

See the table in the next section for a description of the file transfer formats.

3. Check the local directory when in FTP:

```
FTP>LDIR
```

4. Use the LCD or SET DEFAULT /LOCAL command to move to other directories on the local host.

See the DIRECTORY, LDIR, and SET DEFAULT commands in the *Command Reference* for checking directories.

```
FTP> DIRECTORY      [1]
total 49
-rwxr-xr-x    1  smith    users     340   Oct    1   16:34   .login
-rwxr-xr-x    1  smith    users     138   Oct    1   16:34   .profile
drwxr-xr-x     2  smith   users     512   Oct    1   16:34   bin          [2]
-rw-r--r--     1  smith   users   46080   Oct    1   10:58   sys.exe
drwxr-xr-x     2  root   daemon     512   Feb  10   2001   wastebasket
FTP> LDIR      [3]
Directory DOC$DISK:[DOC.ENG]
ANDY.TXT;1          CYN.PS;2        DO_HELP.TXT;1
GLOSSARY.TXT;1      HELP.DIR;1      KIT_INFO.PS;1
LWK_PERSONAL_LINKBASE;1          SCREEN-FTP.DIR;1    SEND-NORM.1C;
Total of 9 files.

FTP> LCD [.HELP]      [4]
FTP> LDIR
Directory DOC$DISK:[DOC.ENG.HELP]
BUILD.COM;1          FTPHELP.HLB;2          FTPHELP.RNO;1      F
FTPHELP.RNO;1        HELP.MMS;1

Total of 6 files.
```

Checking File Transfer Formats

You can determine what file format to use during file transfers. The FTP client lets you transfer files in formatted ASCII, formatted binary, image, block, FORTRAN carriage control, and VMS formats. On OpenVMS systems, the filename extension can indicate the file type. Formatted ASCII is the default transfer file type and is usually sufficient for most files.

FTP converts the various file formats to formatted ASCII or IMAGE. (Executable and zip/compressed files are popular files in this category.) The formats are similar to the formats that the OpenVMS EXCHANGE utility provides to transfer between OpenVMS and DOS-11 or RT-11 file systems. You either specify the file transfer format when you use the `GET`, `PUT`, or `COPY` command, or the FTP client determines the format from the source filename's extension.

See below for an explanation of the file transfer formats.

Check file extensions to determine file types. You might need to enter special qualifiers when you transfer certain types of files.

When you use the `COPY`, `GET`, or `PUT` commands to transfer files, you can use the `/ASCII`, `/BINARY`, `/BLOCK`, `/FORTRAN`, `/IMAGE`, or `/VMS` qualifiers to set the file transfer format. You can also set default file transfer formats using these qualifiers with the `SET DEFAULT` command, or specifying these keywords with the `TYPE` command. (See the `SET DEFAULT` and `TYPE` commands in the *Command Reference* for equivalent usage.)

This file format...	**With extension...**	**Means...**
Formatted ASCII		ASCII records terminated with a CR and LF and transferred as ASCII. Use for all except formatted binary and image files. Maximum formatted ASCII record size is 8192 bytes. In OpenVMS-to-"FTP ASCII" conversion, CR/LF pairs are added to the end of records. In "FTP ASCII"-to-OpenVMS conversion, CR/LF pairs are removed from the end of records.

Formatted Binary	`.OBJ`
	`.STB`
	`.BIN`
	`.LDA`
		Binary records transferred as IMAGE. In OpenVMS-to-"FTP IMAGE" conversion, record header and checksum are added to all records. In "FTP IMAGE"-to-OpenVMS conversion, record header and checksum are removed from each record.

		Remote hosts might not be able to distinguish between formatted binary and image files because both file types are transferred using "FTP IMAGE" format. In this case, the formatted binary files are stored as image files (and if properly transferred back, are formatted binary files again). This is typically not a problem because formatted binary files are system-dependent files.
BLOCK		File blocks transferred as IMAGE. Use for STREAM, STREAM_CR, STREAM_LF, and UNDEFINED record formats. Provides the highest transfer rates since it involves minimal processing. Very similar to image mode. In OpenVMS-to-"FTP IMAGE" conversion, and OpenVMS file is read using block-I/O mode without regard to record structure. In "FTP IMAGE"-to-OpenVMS conversion, an OpenVMS file is created with the STREAM_LF record format and is written using block-I/O mode. No padding of the last block of data occurs. Block mode is particularly useful for files with a STREAM, STREAM_CR, STREAM_LF, or UNDEFINED record format.
FORTRAN		Like formatted ASCII except that first character of each line controls how to display each line. Conversions are the same as for formatted ASCII. Attributes for the output file reflect that the file has a FORTRAN carriage control format. Some hosts do not distinguish between FORTRAN carriage control and ASCII files and might not support this transfer format.
IMAGE	.EXE .TSK .OLB .MLB .SYS .SML .ULB	Fixed-length binary records transferred as IMAGE. In OpenVMS-to-"FTP IMAGE" conversion, records are read as is. In "FTP IMAGE"-to-OpenVMS conversion, records are written as fixed length. If the last record is too short (less than 512 bytes), it is padded with binary zeros.

VMS		Use for RMS file transfers between OpenVMS systems. Systems that support this structure negotiate it automatically. The VMS file structure types are richer than those of UNIX for which FTP is designed. Thus, VMS and VMS-Plus modes were added to help in transferring OpenVMS files.

Using GET, PUT, and COPY

Use the `GET`, `PUT`, or `COPY` commands to transfer files.

`GET`	"Gets" a copy of a file from the remote host and places it in the current local directory.
`PUT`	"Puts" a copy of a local file in the current directory on the remote host.
`COPY`	"Gets" or "puts" a copy of a file, depending on use of the `/LOCAL` or `/REMOTE` qualifier after the source or destination parameter. `COPY` requires the destination parameter.

The below example shows the format and filename syntax of the `GET`, `PUT`, and `COPY` commands. Follow the examples and observe the following conventions when you transfer files between remote and local hosts (the sequence is not important):

- If using `GET` or `PUT`, omit *`destination`* if you want to use the *`source`* filename (and extension if it exists), unless *`source`* is a quoted string. `COPY` requires the destination parameter. If using `COPY`, use a wildcard (asterisk) for *`destination`* when you want to use the source filename as the destination filename.
- If copying to or from a non-OpenVMS file specification, enclose it in double quotes (`" "`).
- Separate multiple file specifications with commas.
- If using wildcarded source file specifications (with an asterisk), use the `/MULTIPLE` qualifier. Alternatively, use the `MGET` or `MPUT` command to copy wildcarded source files. (Note that this requires setting the remote default directory first.)
- Including an asterisk (`*`) after the semicolon (`;`) in a destination parameter preserves the file version when copying to a remote host. Note that if the file version in the source

parameter already exists at the destination, that version is overwritten at the destination. Also, you do not get a warning if a higher numbered destination version already exists.

- If a DECnet file, use the full OpenVMS file specification.
- At this point, the file transfer format you determined is important.

See the `GET`, `PUT`, and `COPY` commands in the *Command Reference*.

> **Note:** TCPware FTP does fast transfers between two OpenVMS systems using VMS file structure or VMS Plus Mode (for TCP/IP Services for OpenVMS (UCX) servers). When TCPware FTP identifies file transfers between two OpenVMS hosts running TCPware, it automatically transfers files in large blocks rather than small records. These VMS modes greatly increase the transfer speed and preserve all Record Management Services (RMS) file attributes. The VMS modes are disabled with non-OpenVMS systems.

```
$ CREATE FTP_STARTUP.COM
OPEN IRIS SMITH "Sandy"
SHOW STATUS
<Ctrl/z>
$ EDIT LOGIN.COM
.
.
.
$ DEFINE/PROCESS FTP_STARTUP "SYS$LOGIN:FTP_STARTUP.COM"
<Ctrl/z>

$ FTP
220 IRIS.process.com (192.168.12.34) FTP-OpenVMS FTPD V5.5 (c)  2001
Process Software
331 Password required.
230-
230-       Welcome to OpenVMS VAX V6.2 (IRIS)
230-               with TCPware  5.5
230-
230 User logged in, proceed.
257 "SYS$SYSROOT:[SYSMGR]"
Client-FTP  V5.5  Copyright (c) 2001 Process Software

Connected to IRIS.process.com (192.168.12.34).
Logged in as user "SMITH".

The local default is ENG_DOC:[ENGINEERING.SMITH]
The remote working directory is SYS$IRIS:[SMITH]

Default qualifiers are /VMS

FTP>
```

Anonymous Users

You can access some remote resources as an ANONYMOUS user instead of with your usual username and password. This is especially useful for access to sites that allow anonymous user access to some of their files.

Anonymous access depends on your use of the /ANONYMOUS qualifier with the FTP commands that require a file or directory specification using the node name syntax.

You can access some remote resources as an ANONYMOUS user in one of the following ways (see the below example):

1. By default, use the node name file syntax (as described below) with any FTP command that requires a file or directory specification (such as COPY, DIRECTORY, RENAME, and SET DEFAULT). This file syntax sends the ANONYMOUS username and your e-mail address as a password.

   Thus, the following file or directory specification: *node*::*path*

   is equivalent to: *node*"ANONYMOUS *your-email-address*"::*path*

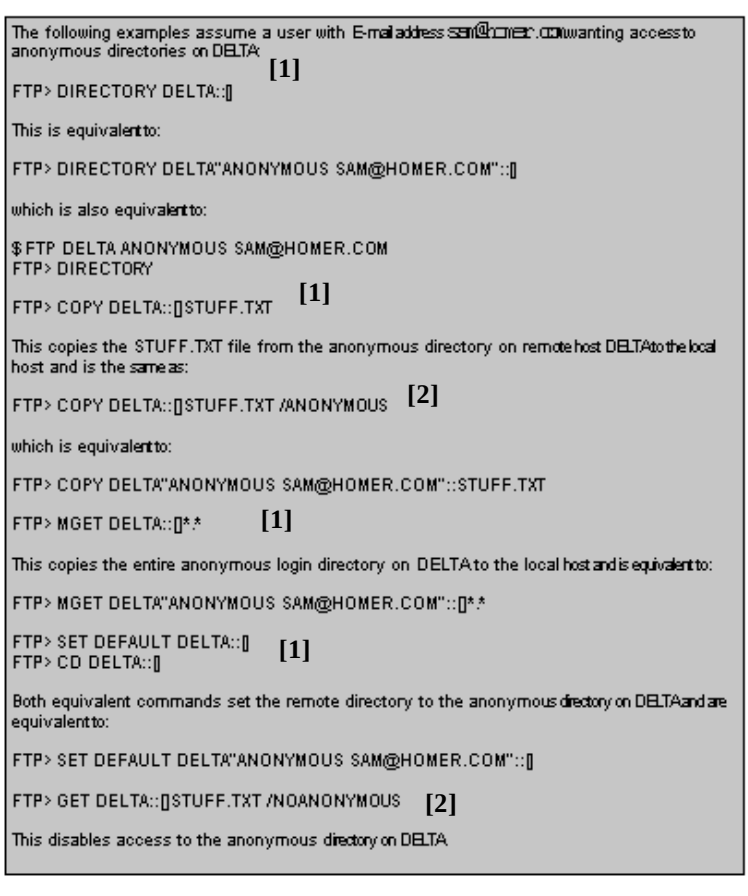
   With OpenVMS V6.1 and later on Alpha and Itanium systems, node can be a domain name or IP address.

2. Use the file specification syntax described in the below example and (optionally) add the /ANONYMOUS qualifier or deny remote anonymous access using the /NOANONYMOUS qualifier.

Using the node name file syntax (and the /ANONYMOUS or /NOANONYMOUS qualifier) affects the following FTP commands:

COPY	CREATE/DIRECTORY	DELETE	DIRECTORY	DISPLAY
GET	LS	MDELETE	MGET	MKDIR
MPUT	PUT	RENAME	RMDIR	SET DEFAULT

Below are examples of how to allow or deny anonymous user access to remote resources.



The following examples assume a user with E-mail address sam@homer.com wanting access to anonymous directories on DELTA:

FTP> DIRECTORY DELTA::[] **[1]**

This is equivalent to:

FTP> DIRECTORY DELTA"ANONYMOUS SAM@HOMER.COM"::[]

which is also equivalent to:

$ FTP DELTA ANONYMOUS SAM@HOMER.COM
FTP> DIRECTORY

FTP> COPY DELTA::[]STUFF.TXT **[1]**

This copies the STUFF.TXT file from the anonymous directory on remote host DELTA to the local host and is the same as:

FTP> COPY DELTA::[]STUFF.TXT /ANONYMOUS **[2]**

which is equivalent to:

FTP> COPY DELTA"ANONYMOUS SAM@HOMER.COM"::STUFF.TXT

FTP> MGET DELTA::[]*.* **[1]**

This copies the entire anonymous login directory on DELTA to the local host and is equivalent to:

FTP> MGET DELTA"ANONYMOUS SAM@HOMER.COM"::[]*.*

FTP> SET DEFAULT DELTA::[]
FTP> CD DELTA::[] **[1]**

Both equivalent commands set the remote directory to the anonymous directory on DELTA and are equivalent to:

FTP> SET DEFAULT DELTA"ANONYMOUS SAM@HOMER.COM"::[]

FTP> GET DELTA::[]STUFF.TXT /NOANONYMOUS **[2]**

This disables access to the anonymous directory on DELTA

Startup Command File

You can have a startup file execute FTP commands each time you invoke FTP. The startup file contains commands you want your system to perform at the beginning of each FTP session. Your system manager might already have defined a system-wide FTP startup file. Creating an FTP startup file is optional.

The startup command file in the below example opens a remote connection, sends the password, and initiates a `SHOW STATUS` command.

You can set up an FTP startup command file or override one established by the system manager at the system level using the following procedure:

1. Create an `FTP_STARTUP.COM` file in your directory.
2. In the file, include the FTP commands you want executed each time you start an FTP session. If you include a password, make sure to use quotation marks to preserve case.

3. Edit your LOGIN.COM file and define the FTP_STARTUP logical to point to the startup file:

```
$ DEFINE/PROCESS FTP_STARTUP "SYS$LOGIN:FTP_STARTUP.COM"
```

Using the DEFINE/PROCESS FTP_STARTUP entry in the user's LOGIN.COM file causes that file to override any FTP startup command file at the system level.

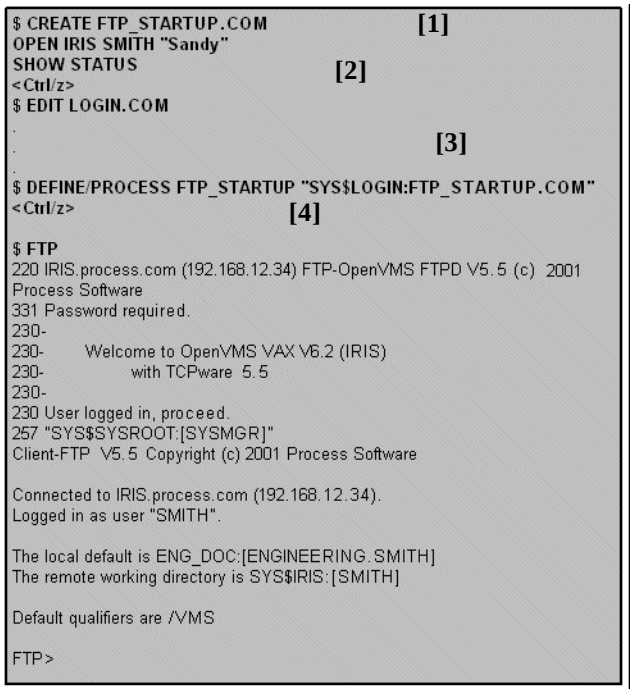
4. Run FTP.

Whenever you run the TCPware FTP client, it looks for the file to which the FTP_STARTUP logical points, and processes all the commands in that file.

If the EXIT or QUIT command appears in the startup file, the FTP client:

- Ignores all commands following the EXIT or QUIT command.
- Continues with FTP operations after the startup command file.

Note: VERBOSE mode is set ON by default so that you can read replies from the FTP server when you connect or change server directories. This means that you do not need to include the SET DEBUG /CLASS=REPLIES (or its equivalent VERBOSE) command in the startup command file. Although an existing SET DEBUG /CLASS=REPLIES command in the file does not change the mode, a VERBOSE command toggles VERBOSE mode OFF. (See the SET DEBUG /CLASS command description in the *Command Reference*.) If you are an ANONYMOUS user, VERBOSE mode might help in reading any informational messages the FTP server creates.



Site-Specific Commands

The TCPware FTP server supports the `SITE SPAWN` and `SITE SHOW TIME` site-specific commands. The FTP client can issue these commands at any time.

Site-specific commands can vary depending on the remote FTP server; some servers do not support any.

Issue the TCPware FTP site-specific commands in one of the following ways at the `FTP>` prompt (see the below example):

1. This command returns the current date and the time of day for the OpenVMS system in the reply message.

```
FTP> SITE SHOW TIME
```

2. This command allows you to execute any DCL command as a subprocess. You typically use this command to print files, submit batch jobs, execute command procedures, or issue other commands.

```
FTP> SITE SPAWN dcl-command
```

The screen does not display the output the subprocess generates. The system returns status from the subprocess as the status for the SITE SPAWN command. Note that spawning is not allowed for CAPTIVE accounts. See the SITE and SPAWN commands in the *Command Reference*.

```
$ FTP
FTP> OPEN CONDOR
_Username [wombat]:WOMBAT
_Password:
FTP> SITE SHOW TIME                                                    [1]
200 The date and time is "3-NOV-2001 11:56:18.
FTP>

FTP> DIR
Directory DOC$DISK:[DOCUMENT.WOMBAT]

ANDY.TXT;1          4     4-NOV-2001    09:08:41.13
CYN.PS;2           53    14-JAN-2001    14:10:41.22
DNIP.TXT;1          8    10-JAN-2001    14:00:08.40
DO_HELP.TXT;1       8    19-NOV-2001    09:49:37.92

FTP> SITE SPAWN PRINT/QUE=ENG_PRINTER_ANSI ANDY.TXT                    [2]
200 SITE command okay.
FTP>
```

Sample Session

This section describes a sample TCPware FTP session.

See the below example for the corresponding numbered steps. In this example, a user on local host BETA:

1. Starts the TCPware FTP client, opens a connection to remote host THETA, and logs in as user SMITH (the display does not echo the password at the prompt).
2. Using PUT, copies the local SYS.EXE file to THETA.
3. Using GET, copies the SYS.EXE file on THETA back to BETA.
4. Obtains a remote directory listing. There is a SYS.EXE file.
5. Deletes the SYS.EXE file.
6. Obtains another remote directory listing. SYS.EXE is now gone.
7. Obtains a local directory listing. Note that SYS.EXE;1 still exists locally.
8. Opens a connection to host ALPHA (running OpenVMS and the TCPware FTP server) and logs in as USER. This closes the connection to THETA.
9. Obtains a remote directory listing on ALPHA.
10. Using GET, copies the ASCII file SCREEN_FTP.TXT on ALPHA to BETA.
11. Changes the default for transferring files from formatted ASCII to IMAGE.
12. Using GET, copies the SEND-NORM.BIN, SEND-NORM.OBJ and SEND.OBJ files from ALPHA as image files on the local host.
13. Obtains a local directory listing. SCREEN-FTP.TXT, SEND-NORM.BIN, SEND-NORM.OBJ, and SEND.OBJ are now present.

14. Exits FTP.

```
(BETA) $ FTP
FTP> OPEN THETA                                    [1]
_Username [smith]: SMITH
_Password:
FTP> PUT SYS.EXE*                                  [2]
FTP> GET SYS.EXE*                                  [3]
FTP> DIR                                           [4]
total 4
-rwxr-xr-x      1       smith   users   340     Oct     1       16:34   .login
-rwxr-xr-x      1       smith   users   138     Oct     1       16:34   .profile
drwxr-xr-x      2       smith   users   512     Oct     1       16:34   bin
-rw-r--r--      1       smith   users   46080   Oct     1       10:58   sys.exe
FTP> DELETE SYS.EXE                                [5]
FTP> DIR                                           [6]
total 3
-rwxr-xr-x      1       smith   users   340     Oct     1       16:34   .login
-rwxr-xr-x      1       smith   users   138     Oct     1       16:34   .profile
drwxr-xr-x      2       smith   users   512     Oct     1       16:34   bin
FTP> LDIR                                          [7]
Directory DOC$DISK:[DOC.ENG]

ANDY.TXT;1              CYN.PS2         DO_HELP.TXT;1
GLOSSARY.TXT;1          HELP.DIR;1      KIT_INFO.PS;1
LWK_PERSONAL.LINKBASE;1                 SYS.EXE;1

Total of 8 files.
FTP> OPEN ALPHA
_Username [smith]: USER                            [8]
_Password:
FTP> DIR                                           [9]
GLOSSARY.TXT;1          HOME.DIR;1              KIT_BUILD.HLB;1
LWK_PERSONAL.LINKBASE;1 SCREEN-FTP.TXT;1        SEND-NORM.BIN;1
SEND-NORM.OBJ;1         SEND.OBJ;1

FTP> GET SCREEN-FTP.TXT                            [10]
FTP> SET DEFAULT /IMAGE                            [11]
FTP> GET SEND-NORM.BIN, SEND_NORM.OBJ,SEND.OBJ                  [12]
FTP> LDIR                                          [13]
Directory DOC$DISK:[DOC.ENG]

ANDY.TXT;1              CYN.PS2             DO_HELP.TXT;1
GLOSSARY.TXT;1          HELP.DIR;1          KIT_INFO.PS;1
LWK_PERSONAL.LINKBASE;1                     SCREEN-FTP.TXT;1
SEND-NORM.BIN;1         SEND-NORM.OBJ;1       SEND.OBJ;1
SYS.EXE
Total of 12 files.
FTP> EXIT                                          [14]
```

Command Reference

The following pages describe the TCPware FTP commands. The immediately following table contains command synonyms you can use interchangeably with TCPware FTP commands. The second following table shows commands you can use to do various tasks.

Enter FTP commands at the `FTP>` prompt. The TCPware FTP client supports the following commands:

ACCOUNT	ENABLE VMS_PL	PWD	SET PASSIVE

CCC	ERROR_EXIT	QUOTE	SET VMS
CLOSE	EXIT	REMOTEHELP	SET STATUS
COPY	GET	RENAME	SITE
CREATE/DIR	HELP	SET BELL	SPAWN
DEFINE/KEY	LDIR	SET DEBUG	STRUCTURE
DELETE	OPEN	SET DEFAULT	TYPE
DIRECTORY	PROTECTION	SET HASH	USER
DISPLAY	PUT	SET LOWERCASE	

FTP client command synonyms:

This command...	**Is a synonym for the FTP command...**
ASCII	TYPE ASCII
BELL	Toggles between SET BELL and SET NOBELL
BINARY or IMAGE	TYPE IMAGE
BYE or QUIT	EXIT
CD	SET DEFAULT /REMOTE
CONNECT	OPEN
DEBUG	Toggles SET DEBUG/CLASS=COMMANDS
DISCONNECT	CLOSE
H	HELP

HASH	Toggles between SET HASH and SETNOHASH
LCD	SET DEFAULT/LOCAL
LIST or LS	DIRECTORY/NAME_LIST
LOGIN	USER
MDELETE	DELETE/MULTIPLE
MGET	GET/MULTIPLE
MKDIR	CREATE/DIRECTCORY
MPUT	PUT/MULTIPLE
PASSIVE	Toggles between SET PASSIVE and SET NOPASSIVE
RECV	GET
RM	DELETE
RMDIR	DLETE/DIRECTORY
SEND	PUT
STATUS	HOW STATUS
VERBOSE	Toggles SET DEBUG/CLASS=REPLIES
Z	SPAWN

Commands used to perform tasks on the local system:

DEFINE/KEY	Associate an equivalence string and set of attributes with a keyboard key
HELP	Bring up the FTP client online help facility

LCD	Set your local default directory
LDIR	List files in your local directory
SET BELL	Ring terminal bell after completing a file transfer
SET DEBUG	Display of debugging information
SET DEFLATE	Sets DEFLATE mode optional parameters. The only optional parameter currently recognized is /LEVEL, which can be specified as -1 (default, balance between compression and CPU time), 0 (no compression) to 9 (maximum compression). The SET MODE DEFLATE command must be used to enter deflate (ZLIB compression) mode. DEFLATE mode is not compatible with TLS authentication, which provides its own data compression algorithms. DEFLATE mode cannot be used when TLS is being used.
SET HASH	Enable hash marks during a file transfer
SET LOWERCASE	Convert unquoted filenames to lowercase in a file transfer request
SET PASSIVE	Sets passive mode
SET VMS	The FTP client negotiates with the server for VMS file structure when opening a connection
SHOW STATUS	Show the status of the current connection and local default directory
SPAWN	Executive DCL commands without exiting FTP
STRUCTURE	Change the default file structure for a transfer (FILE, RECORD, or VMS)
TYPE	Change the default file transfer format (ASCII, BINARY, IMAGE, FORTRAN, BLOCK, VARIABLE, or DEFAULT)

Commands used to perform tasks on remote systems:

CD	Change the remote default directory
DELETE	Delete a file or directory on the remote host
DIR, LIST, or ls	List files on the remote host
MKDIR	Create a directory on the remote host
PWD	Display the name of the current working directory on the remote host
QUOTE	Send an FTP command to the remote server
REMOTEHELP	Bring up the remote FTP server's online help facility
RENAME	Rename a file on the remote host
SITE	Issue a site-specific command to the remote server
USER	Set the username at the remote host

TCPware FTP logicals for users:

FTP_STARTUP
Define the FTP_STARTUP logical to point to the FTP_STARTUP.COM file. For example:

```
$ DEFINE /SYSTEM/EXECUTIVE 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 system wide setting.

TCPWARE_FTP_MAX_PRE_ALLOCATION
The logical TCPWARE_FTP_MAX_PRE_ALLOCATION may be defined to limit the size that a file will be pre-allocated to when file size information is available at transfer time. This can be

important when transferring very large files, as it can take a long time to pre-allocate the file at the start of the transfer and timeout routines in FTP and/or firewalls may cause connections to be dropped. This logical does not have any effect for STRU OVMS transfers of Indexed, Contiguous, or Contiguous, Best Try files; these files need to have accurate allocation size information at the start of the transfer.

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 TCPWARE_FTP_STRIP_VERSION has no effect.

TCPWARE_FTP_ALL_VERSIONS is ignored if the FTP server is in UNIX emulation mode.

TCPWARE_FTP_DISALLOW_UNIX_STYLE
Controls whether UNIX style filename parsing is done. If not defined and a / is found in the filename, it is assumed to be a UNIX style filename.

TCPWARE_FTP_EXTENSION_QUANTITY
Defines the default allocation/extension quantity for new files and appends.

TCPWARE_FTP_IGNORE_UNIX_DASH_OPTIONS
By default, the FTP server ignores Unix-style dash options on LIST and NLST when in Unix mode (for example, the "-l" in "ls -l"). Define this to be FALSE to tell the FTP server to pay attention to Unix-style dash options.

TCPWARE_FTP_ONLY_BREAK_ON_CRLF
If this 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_SEMANTICS_FIXED_IGNORE_CC
If this 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.

TCPWARE_FTP_SEND_FEAT_ON_CONNECT

By default, the FTP client sends the `FEAT` command upon connecting to a server. This can be disabled by defining this logical as `FALSE`. When this is disabled the FTP client will not be able to detect the support of optional features such as `TLS`, `REST STREAM`, and others and these features may not work correctly if there is an attempt to use them.
`TCPWARE_FTP_STRIP_VERSION` Causes VMS mode output to have no versions.
`TCPWARE_FTP_USE_SRI_ENCODING_ON_ODS5` The logical `TCPWARE_FTP_USE_SRI_ENCODING_ON_ODS5` can be defined to `1`, `TRUE` or `YES` to cause the file name encoding used for UNIX-style file names on ODS-2 disks to be used on ODS-5 disks. This also sets the default case of letters in filenames to lowercase and ignores the stored case.
`TCPWARE_FTP_UNIX_STYLE_CASE_INSENSITIVE` Allows UNIX style filename handling to be case insensitive.

Troubleshooting

Access error messages help by entering `HELP TCPWARE MESSAGES [`*`identifier`*`]`, or visit the Process Software web site at www.process.com.

ACCOUNT

Specifies the user's account if the remote server requires it.

Format

```
ACCOUNT account
```

Parameter

account
User's account. Enclose in quotes if it contains special characters or embedded spaces, or contains mixed-case characters.

Example

The following specifies account Smith on the remote system. Use quotes around the mixed-case account name.

```
FTP>ACCOUNT "Smith"
```

CCC

Change the control port to clear text after performing RFC 4217 encrypted authentication. Clear text may be desired for the control port when NAT or firewalls are being used that expect to examine and/or alter commands and responses dealing with the data port (`PORT`, `PASV`, `EPRT`, `EPSV` and the respective replies). The `PROTECTION` command should be used before the `CCC` command as it is not allowed after the command channel has returned to clear text mode.

Format

```
CCC
```

CLOSE

Closes the connection to the remote FTP server if one is open and keeps you in FTP.

OPEN and CONNECT also close an existing connection before opening another one.

Format

CLOSE

Synonym

DISCONNECT

Example

The following closes the current connection:

```
FTP>CLOSE
```

COPY

Copies files to or from a remote host. You specify whether the source or destination file is local or remote using the /LOCAL or /REMOTE qualifier. COPY supports full wildcard file specifications except wildcard symbols enclosed in a quoted string. Use the /MULTIPLE qualifier for a wildcard remote source file specification. /REMOTE also supports use of asterisk (*) wildcards after a semicolon (;) in remote file specifications. This creates the same version in the destination file as in the source file (instead of creating a new version). If the server is not OpenVMS, the version number is part of the filename. TCPware does not issue a warning if the server host already has a higher numbered version.

Format

COPY *source* [,*source*,...] *destination*

Equivalents

GET = COPY *source* /REMOTE *destination*
RECV = COPY *source* /REMOTE *destination*
MGET = COPY *source* /REMOTE /MULTIPLE *destination*
PUT = COPY *source* /LOCAL *destination*
SEND = COPY *source* /LOCAL *destination*
MPUT = COPY *source* /LOCAL /MULTIPLE *destination*

Parameters

source

Input file specification. Use a comma between multiple file specifications. Enclose the file specifications in quotes if you want to preserve case and did not use the SET NOLOWERCASE command.

destination

Output file specification. Enclose the file specification in quotes if you want to preserve case and did not use the SET NOLOWERCASE command. If wildcarded (*), the FTP client uses the source filename or extension, unless the file specification is a quoted string. See the source parameter for the destination file specification format.

To obtain the same version number in the destination file as in the source file (instead of creating a newer one), wildcard the destination file version using ;*. Note that if the server is not an

OpenVMS host, the version number is included in the filename. You do not get a warning if the server host already has a higher numbered version. Also, if the server host already has the version specified, the old file with that version is overwritten.

Transfer Qualifiers (Positional)

/LOCAL
The preceding file is on the local host. If /LOCAL follows source, /REMOTE is implicit for *destination*. If /LOCAL is omitted, the FTP client searches for a node; if found, the FTP client assumes the file is remote. Do not use for both source and *destination*.

/REMOTE
The preceding file is on the remote host. If /REMOTE follows *source*, /LOCAL is implicit for *destination*. If /REMOTE is omitted, the FTP client searches for a *node*; if found, the FTP client assumes the file is remote. Do not use for both *source* and *destination*. (See the destination parameter on how to preserve version numbers on a remote copy.)

/MULTIPLE
Transfers multiple files. Use after *source* only. Include wildcards in *source* only because some remote hosts do not recognize the OpenVMS asterisk and percent characters as wildcards. The remote host's server must support the FTP NLST command. Not all servers support VMS files. If the server does and you do not specify another mode (using a qualifier or the STRUCTURE or SET DEFAULT commands), /VMS is the default.

File Type Qualifiers (Positional)

If you omit one of the file type qualifiers, the FTP client transfers the file based on either:

- The current default setting; for example, ASCII or IMAGE.
- The extension (type) of the file you want to copy (see *File Transfer Formats*).

Setting a file type qualifier overrides the default transfer format for this transaction only. (See also the SET DEFAULT command.)

/ASCII
Transfers the preceding file in formatted ASCII format.

/BINARY
Transfers the preceding .BIN, .LDA, .OBJ, or .STB file in formatted binary format.

/BLOCK
Transfers the preceding STREAM, STREAM_CR, STREAM_LF, or UNDEFINED file in block mode.

/FORTRAN
Transfers the preceding file in FORTRAN mode. The first character of each record is a FORTRAN carriage control character. Some hosts do not recognize this transfer format.

/IMAGE[=*size*]
Transfers the preceding file in image mode. Optional *size* sets the record size of the local output file. Does not apply to remote output files. The maximum size for this qualifier is 32768.

/RECORD
Transfers the preceding file using STRU R so as to communicate the record structure during the copy. Not all servers support record structure mode. If you specify both /RECORD and /VMS, the FTP client uses /VMS.

/VARIABLE
Transfers an image file (see /IMAGE) in variable length record mode. At the destination site, all /IMAGE records have a fixed length. Applies to local output image files only. This qualifier has meaning only if the /IMAGE qualifier is present.

/VMS
Transfers the preceding file in VMS file mode. Allows you to transfer any type of RMS file between OpenVMS systems. If you use /VMS, the FTP client ignores /APPEND, /ASCII, /BINARY, /BLOCK, /FORTRAN, /IMAGE, and /VARIABLE. If you specify both /RECORD and /VMS, the FTP client uses /VMS.

Other Qualifiers (Non-positional)

/ANONYMOUS
/NOANONYMOUS

Enables (/ANONYMOUS) or denies (/NOANONYMOUS) anonymous user access to remote resources. You can omit /ANONYMOUS if you use the node file syntax (*node*::*pathname*). (See *Anonymous Users*.)

/APPEND
Appends the *source* file to the *destination* file. If the *destination* file does not exist, the FTP client creates it. Only valid if appending to a file with the same file transfer type. Some remote hosts might not support this operation.

/CONFIRM
/NOCONFIRM
/CONFIRM issues a confirmation prompt before copying a file. Useful when *source* contains wildcards so that you can confirm each file copy. Respond with Y or N. /NOCONFIRM is the default.

If confirming multiple file copying, use with COPY/MULTIPLE with a wildcard value. Position the qualifier immediately after the COPY verb to relate to all files, or after the particular filename to relate to that file only.

/CONTIGUOUS=*blocks*
Local output file should have an initial contiguous allocation of the specified number of *blocks*. If the output file is smaller than the specified *blocks*, the FTP client truncates the allocation. If the output file is larger, the additional allocations are non-contiguous. Does not apply to remote output files.

/FDL
Uses and then deletes a separate FDL file describing the specified file's OpenVMS RMS record attributes. This qualifier is useful after a PUT /FDL operation from a VMS node transfers a file to a non-VMS node: the GET /FDL operation can then return the file with the proper record attributes back from the non-VMS node. The default is not to create an accompanying FDL file. The TYPE command determines the type of file. A transfer of:

- ASCII data results in a sequential file with variable length records (the default).
- IMAGE data results in a sequential file with fixed length records of 512 bytes.

/IGNORE
/NOIGNORE
/IGNORE ignores errors so that copying can continue with the next file. /NOIGNORE, the default, terminates copying if an error occurs.

/LOG
/NOLOG
/LOG displays file specifications for each file transferred. /NOLOG, the default, does not display the transferred file's specifications.

/RESTART
For STREAM mode transfers restart the transfer where it was interrupted. The client verifies that the server supports the RFC 3659 SIZE and REST commands, and ignores the qualifier if it does not.

This does NOT work for VMS mode transfers (STRU VMS), and if the remote system is a VMS system it is recommended that a STRU FILE be done before the transfer command and to include /NOVMS on the command line.

/SET_FACTS
Set selected file facts on the destination file to match the source file after transfer. The currently supported fact is MODIFICATION__TIME.

Examples

1. Each of these commands copies the STUFF.TXT file from the local host to remote host SYS1 (the receiving system stores the file under the same filename in user SMITH's directory):

```
FTP>COPY STUFF.TXT SYS1"SMITH SECRET"::
FTP>PUT STUFF.TXT SYS1"SMITH SECRET"::
```

2. Each of these commands copies the DATA1.TXT and DATA2.TXT files from the remote host to the local host, assuming that a connection to the remote host is currently open:

```
FTP>COPY DATA1.TXT,DATA2.TXT /REMOTE *
FTP>GET DATA1.TXT,DATA2.TXT
```

3. Each of the following commands copies all .BAS files from a remote OpenVMS host to the local host. The /MULTIPLE qualifier and the asterisk wildcard are used in the COPY command, and they are omitted in the equivalent MGET command.

```
FTP>COPY *.BAS/REMOTE/MULTIPLE *
FTP>MGET *.BAS
```

4. The issuer of the following command wants to copy all local `.SQL` type files into multiple files in the remote UNIX system's directory.

```
FTP>COPY *.SQL/LOCAL/MULTIPLE "/usr/users/sql/*"
```

To accomplish this, the issuer uses an asterisk wildcard in the output file specification, as in Example 3. However, the result is not as intended. Because the asterisk is part of a quoted string, the command actually copies the files into a single file literally named `*` on the remote host.

To avoid this, set the remote default directory to the full pathname. You do not have to specify the quoted pathname in the `COPY` command:

```
FTP>SET DEFAULT/REMOTE "/usr/users/sql"
FTP>COPY *.SQL/LOCAL/MULTIPLE *
```

The asterisk now acts as a true wildcard, with the intended result.

CREATE/DIRECTORY

Creates a directory on the remote host. The /DIRECTORY qualifier is required as part of the command. Some remote hosts might not support directory creation operations.

Format

CREATE/DIRECTORY *remote-directory*

Synonym

MKDIR

Parameter

remote-directory
Directory to create on the remote host, in the format:

```
[node"username password"::]directory
```

To open a connection first, use the *node*"*username password*"**::** part of the format. This syntax is optional. If you omit the parameter and a connection is already open, the FTP client uses the current default directory. The *directory* part of the format is any valid remote directory specification. Enclose the specification in quotes if it contains special characters or embedded spaces or is case-sensitive.

Use the *node*::*directory* syntax to create an anonymous user directory. The /ANONYMOUS qualifier is implicit.

Qualifier

/ANONYMOUS
/NOANONYMOUS
Enables (/ANONYMOUS) or denies (/NOANONYMOUS) creation of anonymous user directories. You can omit /ANONYMOUS if using the node file syntax (*node*::*pathname*). (See *Anonymous Users*.)

Examples

1. These commands are equivalent and create a directory `USERS` on the remote OpenVMS host SYS1, with the username and password specified explicitly:

```
FTP>CREATE/DIRECTORY SYS1"SMITH SECRET"::[USERS]
FTP>mkdir sys1"smith secret"::[users]
```

2. All three of the following commands create a directory `USERS` in the `anonymous` directory on the remote OpenVMS host SYS2.

```
FTP>CREATE/DIRECTORY SYS2::[USERS]
FTP>mkdir sys2::[users]
FTP>mkdir sys2::[users] /anonymous
```

The commands are equivalent to:

```
FTP>CREATE/DIRECTORY SYS2"ANONYMOUS user-email-address"::[USERS]
```

DEFINE/KEY

Associates an equivalence string and a set of attributes with a key on the terminal keyboard.

Format

`DEFINE/KEY` *`key-name`* `["]`*`equivalence-string`*`["]`

Parameters

`key-name`

Name of the key to define. The below table lists key designations for three terminal types:

- On LK201 terminals, you can define three types of keys: numeric keypad, editing keypad (except the up and down arrow keys), and function key row (except F1 through F5).
- On VT100-type terminals, you can also define the left arrow and right arrow keys. On VT200 terminals, the left arrow and right arrow keys, and the F6 through F14 keys, are for command line editing. Issue the DCL command `SET TERMINAL/NOLINE_EDITING` to define these keys before you run the FTP client. You can also press `Ctrl+V` to enable keys F7 through F14 (but not F6).
- On VT52 terminals, the only definable keys are on the numeric keypad.

Key Name	LK201	VT100-type	VT52
PF1	PF1	PF1	[blue]
PF2	PF2	PF2	[red]
PF3	PF3	PF3	[gray]
PF4	PF4	PF4	
KP0,...,KP9	0,...,9	0,...9	0,...9
PERIOD	.	.	.
COMMA	,	,	,
MINUS	-	-	-

ENTER	ENTER	ENTER	ENTER
LEFT	←	←	←
RIGHT	→	→	→
Find (E1)	Find		
Insert Here (E2)	Insert_Here		
Remove (E3)	Remove		
Select (E4)	Select		
Prev Screen (E5)	Prev_Screen		
Next Screen (E6)	Next_Screen		
HELP	Help		
DO	Do		
F6,...,F20	F6,...,F20		

equivalence-string
String to substitute when you press the key. If the string contains spaces, enclose it in quotes.

Qualifiers

/ECHO
/NOECHO
/ECHO displays the equivalence string on your screen after you press the key. /NOECHO is the default. Do not use /NOECHO with /NOTERMINATE.

/IF_STATE=(state-name,...)
/NOIF_STATE
/IF_STATE specifies a list of one or more *state-names* (an alphanumeric string) for the key definition to be in effect. If you specify only one *state-name*, you can omit the parentheses.

By including several *state-names*, you can define a key to have the same function in all the specified states. /NOIF_STATE is the default, where the FTP client uses the current state.

Establish states using /SET_STATE.

/LOCK_STATE
/NOLOCK_STATE
/LOCK_STATE specifies that the state set by /SET_STATE remains in effect until explicitly changed. /NOLOCK_STATE is the default, meaning the state which has been set in effect by /SET_STATE is in effect only for the next definable key you press or the next read-terminating character you type.

You can specify /LOCK_STATE only on the same command line as /SET_STATE.

/SET_STATE=*state-name*
/NOSET_STATE
/SET_STATE specifies the *state-name* (an alphanumeric string) you want set for the key. The default is /NOSET_STATE, where the current state locked by /LOCK_STATE is in effect.

/TERMINATE
/NOTERMINATE
/TERMINATE specifies that the FTP client terminates (effectively executes) the current equivalence string when someone presses the defined key. /NOTERMINATE, the default, allows you to create key definitions that insert text into command lines, after prompts, or into other typed text.

Example

The following sets the F1 key on the keyboard to the ""SMITH SECRET"::[USERS]" string, sets the state to 1, and locks the state for that definition:

```
FTP>DEFINE/KEY F1 """SMITH SECRET""::[USERS]" /SET=1 /LOCK
```

DELETE

Deletes files or directories on the remote host.

Format

```
DELETE file[,file,...]
```

Synonyms

```
RMDIR dir[,dir,...] = DELETE /DIRECTORY
MDELETE file[,file,...] = DELETE /MULTIPLE
```

> **CAUTION!** The DIRECTORY command does not list hidden files (files that start with a period). Using any wildcards with the MDELETE command deletes hidden files, which you might need.

Parameters

file
dir
Remote files or directories to delete. When deleting files, *file* can contain wildcards. See the /MULTIPLE qualifier.

Qualifiers

/ANONYMOUS
/NOANONYMOUS
Enables (/ANONYMOUS) or denies (/NOANONYMOUS) deletion of anonymous files or directories. You can omit /ANONYMOUS if using the node file syntax (*node*::*path*). (See *Anonymous Users*.)

/CONFIRM
/NOCONFIRM
/CONFIRM issues a confirmation prompt before deleting a file. Useful when source contains wildcards so that you can confirm each file copy. Respond with Y or N. /NOCONFIRM is the default.

If confirming multiple file deletions, use with MDELETE or DELETE/MULTIPLE with a wildcard value. Position the qualifier immediately after the DELETE verb to relate to all files, or after the filename to relate to that file only.

/DIRECTORY
Deletes a directory (equivalent to RMDIR). If omitted, the FTP client deletes a file. Do not use with /MULTIPLE.

/IGNORE
/NOIGNORE
/IGNORE ignores errors so that deletion can continue with the next file when using /MULTIPLE. /NOIGNORE, the default, terminates the deletion operation if an error occurs.

/LOG
/LOG displays file specifications for each file deleted.

/MULTIPLE
Deletes multiple files (equivalent to MDELETE). You must include wildcards in the file specification. /MULTIPLE is necessary because other systems do not universally recognize the OpenVMS asterisk and percent characters as wildcards. (You do not need this qualifier with multiple deletes between OpenVMS systems.) The remote host's FTP server must support the FTP NLST command for remote wildcard operations to work. Do not use with /DIRECTORY.

Examples

1. The following deletes the proj1 file from the UNIX /usr/src/directory:

```
FTP>DELETE "/usr/src/proj1"
```

2. The following deletes all files with the .TMP extension in the remote default directory. You do not need /MULTIPLE when doing this delete operation between OpenVMS systems. If several versions of any *.TMP file exist, it deletes only the latest version.

```
FTP>DELETE *.TMP/MULTIPLE
```

3. The following deletes all files with the FOO filename in the remote default directory. You do not need /MULTIPLE when doing this delete operation between OpenVMS systems. If several versions of any FOO.* file exist, it deletes only the latest version.

```
FTP>DELETE FOO.*/MULTIPLE
```

4. The following deletes all files and file versions with the FOO filename in the remote default directory. For example, this command deletes FOO.EXE;1, FOO.EXE;2, FOO.C;1, FOO.C;2, and FOO.TXT;1. You do not need /MULTIPLE when doing this delete operation between OpenVMS systems.

```
FTP>DELETE FOO.*;*/MULTIPLE
```

DIRECTORY

Lists files on the remote host. If the remote host is a TCPware host, also lists the creation date and file type.

See LDIR to list files on the local host.

Format

DIRECTORY [*directory*]

Synonym

LS [*directory*]= DIRECTORY {/BRIEF | /NAME_LIST}

Parameter

directory
Directory to list on the remote host.

Qualifiers

/ANONYMOUS
/NOANONYMOUS
Enables (/ANONYMOUS) or denies (/NOANONYMOUS) anonymous user access to remote resources. You can omit /ANONYMOUS if using the directory syntax *node*::*directory*. (See *Anonymous Users*.)

/BRIEF
/NAME_LIST
Returns a list of filenames instead of a normal directory listing (equivalent to LS). Uses the FTP NLST command. /BRIEF and /NAME_LIST are synonyms.

/OUTPUT=*file*
File specification for a local file to receive the directory listing. If omitted, the directory is displayed on your terminal.

Examples

1. The following returns a listing for the remote `/usr/src/` UNIX directory, assuming that a connection to the remote host is open:

```
FTP>DIRECTORY "/usr/src/"
```

2. The following returns a listing for the remote `SYS$SYSTEM` directory, assuming that a connection to the remote host is open:

```
FTP>DIRECTORY SYS$SYSTEM:
```

DISPLAY

Displays a remote file on the screen.

Equivalent to the GET (or COPY /REMOTE) command with SYS$OUTPUT as the local file specification.

If a VMS Plus mode transfer is requested, DISPLAY temporarily cancels VMS Plus mode, transfers the file(s), and resets VMS Plus mode again.

Note that displaying a non-ASCII file might produce unrecognizable output, as would be the case with the DCL TYPE command.

Format

DISPLAY *remote-file*[,*remote-file*,...]

Equivalents

COPY *remote-file*[,*remote-file*,…] /REMOTE [/MULTIPLE] SYS$OUTPUT
[M]GET *remote-file*[,*remote-file*,…] SYS$OUTPUT

Parameters

remote-file
Input file specification on the remote host. Enclose in quotes if you want to preserve case and did not use the SET NOLOWERCASE command, or the file specification contains delimiters or symbols the FTP server can interpret in special ways. Use a comma between multiple file specifications. The remote file specification must conform to the file naming conventions of the remote host.

Examples

The following shows formats of acceptable equivalent commands that implement the DISPLAY function:

```
FTP>DISPLAY TEXT.TXT
FTP>GET TEXT.TXT SYS$OUTPUT
FTP>MGET TEXT.TXT, TEXT2.TXT SYS$OUTPUT
FTP>COPY TEXT.TXT /REMOTE SYS$OUTPUT
```

```
FTP>COPY TEXT.* /REMOTE /MULTIPLE SYS$OUTPUT
FTP>COPY NODE"USER PASSWORD"::TEXT.TXT SYS$OUTPUT
```

ENABLE [/DISABLE] VMS_PLUS

Turns VMS Plus Mode on or off. This lets you specify a transfer mode based on file type, for example, ASCII or image.

In VMS Plus mode, file transfers use File Descriptor Language (FDL) information to create output files.

Format

```
FTP> ENABLE VMS_PLUS
FTP> DISABLE VMS_PLUS
```

ERROR_EXIT

Exits FTP with a specified status if an error occurs in the previous FTP command. This feature is useful when running FTP from a command procedure.

Note that you exit the FTP client if you try to use this command interactively.

Format

```
ERROR_EXIT [status]
```

Parameter

status

Optional status value the DCL $STATUS symbol returns if FTP exits. Specifies which command (or sequence of commands) failed. If omitted, the FTP client uses the status value of the last error.

> **Note:** The FTP client reports the `$STATUS` as the status value ORd with `%X10000000`.

Example

The following example is part of a DCL command procedure:

```
.
$ SET NOON
$ FTP
OPEN LILAC SMITH PASSWORD
ERROR_EXIT %X10000010
PUT DATA_FILE1.TXT
ERROR_EXIT %X10000020
PUT DATA_FILE1.IMG
ERROR_EXIT %X10000030
PUT DATA_FILE1.DES
ERROR_EXIT %X10000040
EXIT
$ FTP_EXIT_STATUS = $STATUS
$ SET ON
$ IF (FTP_EXIT_STATUS .EQ. %X10000010) THEN GOTO LOGIN_FAILED
```

```
$ IF (FTP_EXIT_STATUS .EQ. %X10000020) THEN GOTO TRANSFER_1_FAILED
.
.
```

This command procedure transfers several files and uses `ERROR_EXIT` to detect if any of the transfers fail. `FTP_EXIT_STATUS` returns the following values:

- `%X10000010` if the connection or login to LILAC fails
- `%X10000020` if FTP cannot transfer `DATA_FILE1.TXT`
- and so on
- `1` if the connection is successful

EXIT

Exits FTP and returns to the DCL prompt. If a connection is open, the FTP client closes it before exiting.

Format

```
EXIT
```

Synonyms

```
QUIT
BYE
```

GET

Copies files from a remote host.

GET supports full wildcard file specifications except wildcards enclosed in a quoted string. Use the /MULTIPLE qualifier for a wildcarded remote file specification.

Format

GET *remote-file*[,*remote-file*,...] [*local-filename*]

Equivalents

COPY *remote-file* /REMOTE *local-filename*
MGET *wildcarded-remote-files* = GET *remote-file* /MULTIPLE
RECV *remote-file*[,*remote-file*,...] [*local-filename*]

Parameters

remote-file
Input file specification on the remote host. Enclose in quotes if you want to preserve case and did not use the SET NOLOWERCASE command, or the file specification contains delimiters or symbols the FTP server can interpret in special ways. Use a comma between multiple file specifications.

The remote file specification must conform to the file naming conventions of the remote host. In OpenVMS-to-OpenVMS file transfers, the *remote-file* and *local-filename* formats are the same. (See the *local-filename* parameter).

wildcarded-remote-files
Input file specification on the remote host in wildcarded format. Wildcards include the % or ? symbol to indicate individual characters, and the * symbol to indicate multiple characters. Examples of wildcarded file specifications are *.txt , W????.*, and *.*;*.

local-filename
Output file specification on the local host. If omitted, the FTP client uses the *remote-file* filename (and extension if it exists), unless *remote-file* is a quoted string. If used, must conform to the OpenVMS file name format.

Qualifiers

If you omit one of the file type qualifiers (/ASCII, /BINARY, /FORTRAN, /IMAGE, /VMS), the FTP client transfers the file based on either:

- The current default setting; for example, ASCII or IMAGE.
- The extension (type) of the file you want copied.

Setting a file type qualifier overrides the default transfer format for this transaction only. See also the SET DEFAULT command.

/ANONYMOUS
/NOANONYMOUS
Enables (/ANONYMOUS) or denies (/NOANONYMOUS) anonymous user access to remote resources. You can omit /ANONYMOUS if using the node file syntax (*node*::*path*).

/APPEND
Appends the *remote-file* file to the *local-filename*. If the *local-filename* does not exist, the FTP client creates it. Some remote hosts do not support this operation. **NOTE:** If the operation fails, try appending in binary mode by using the /BINARY qualifier.

/ASCII
Transfers the file in formatted ASCII format.

/BINARY
Transfers .BIN, .LDA, .OBJ, and .STB files in formatted binary.

/BLOCK
Transfers STREAM, STREAM_CR, STREAM_LF, and UNDEFINED files in block.

/CONFIRM
/NOCONFIRM
/CONFIRM issues a confirmation prompt before getting a file. Useful when source contains wildcards so that you can confirm each file copy. Respond with Y or N. /NOCONFIRM is the default.

If confirming multiple file gets, use with MGET or GET/MULTIPLE with a wildcard value. Position the qualifier immediately after the GET verb to relate to all files, or after the particular filename to relate to that file only.

/CONTIGUOUS=*blocks*
Local output file should have an initial contiguous allocation of the specified number of *blocks*. If the output file is smaller than the specified *blocks*, the FTP client truncates the number of blocks allocated. If the output file is larger, the additional allocations are non-contiguous. Does not apply to remote output files.

/FDL
Uses and then deletes a separate FDL file describing the specified file's OpenVMS RMS record attributes. This qualifier is useful after a PUT /FDL operation from a VMS node transfers a file to a non-VMS node: the GET /FDL operation can then return the file with the proper record attributes back from the non-VMS node. The default is not to create an accompanying FDL file. The TYPE command determines the type of file. A transfer of:

- ASCII data results in a sequential file with variable length records (the default).
- IMAGE data results in a sequential file with fixed length records of 512 bytes.

/FORTRAN
Transfers the file in FORTRAN mode. The first character of each record is a FORTRAN carriage control character. Some hosts do not recognize this transfer format.

/IGNORE
/NOIGNORE
/IGNORE ignores errors so that copying can continue with the next file. /NOIGNORE, the default, terminates copying if an error occurs.

/IMAGE[=*size*]
Transfers the file in image mode. Optional *size* sets the record size of the local output file. Does not apply to remote output files.

/LOG
/LOG displays file specifications for each file transferred.

/MULTIPLE

Transfers multiple files (equivalent to MGET). Use after *remote-file* only and include wildcards in *remote-file*. Necessary because some remote hosts do not recognize the OpenVMS asterisk, percent, or question mark characters as wildcards. /MULTIPLE ensures that the remote host understands more than one file is to be transferred. The remote host's server must support the FTP NLST command for remote wildcard operations to work.

/RECORD
Transfers the preceding file using STRU R so as to communicate the record structure during the copy. A positional qualifier. Not all servers support record structure mode. If you specify both /RECORD and /VMS, the FTP client uses /VMS.

/RESTART
For STREAM mode transfers restart the transfer where it was interrupted. The client verifies that the server supports the RFC 3659 SIZE and REST commands, and ignores the qualifier if it does not.

This does NOT work for VMS mode transfers (STRU VMS), and if the remote system is a VMS system it is recommended that a STRU FILE be done before the transfer command and to include /NOVMS on the command line.

/SET_FACTS
Set selected file facts on the destination file to match the source file after transfer. The only fact currently supported is MODIFICATION__TIME.

/VARIABLE
Transfers an image file (see /IMAGE) in variable length record mode. All /IMAGE records are fixed length when stored at the destination. Applies to local output image files only.

/VMS
Transfers the file in VMS file mode. Allows you to transfer any type of RMS file between OpenVMS systems. A positional qualifier. If you use /VMS, the FTP client ignores /APPEND, /ASCII, /BINARY, /BLOCK, /FORTRAN, /IMAGE, and /VARIABLE. If you specify both /RECORD and /VMS, the FTP client uses /VMS.

Not all servers support VMS files. If the server does and you do not specify another mode (using a qualifier or the STRUCTURE or SET DEFAULT commands), /VMS is the default.

Examples

1. The following copies the DATA1.TXT and DATA2.TXT files from the remote host to the local system, assuming that a connection to the remote host is currently open:

```
FTP>GET DATA1.TXT,DATA2.TXT
```

2. The following copies all remote files with extension .BAS from a remote OpenVMS host to the local host:

```
FTP>MGET *.BAS
```

3. The following copies the STUFF.TXT file from DELTA's anonymous directory. It is equivalent to having used /ANONYMOUS. Sends the "ANONYMOUS *user-email-address*" username and password with the command.

```
FTP>RECV DELTA::STUFF.TXT
```

HELP

Accesses the the FTP client online help.

The FTP client help uses the OpenVMS interactive help facility.

To exit the help facility, press **Return** until you return to the FTP> prompt.

See the REMOTEHELP command, or the /REMOTE qualifier, for access to the remote server's online help.

Format

HELP [/REMOTE] [*topic*]

Synonyms and Equivalents

H
REMOTEHELP [*topic*] = HELP /REMOTE [*topic*]
HELP /REMOTE SITE = REMOTEHELP SITE = SITE HELP = QUOTE HELP SITE

Parameter

topic
Optional; allows you to specify the topic, if known, for which you want help. Otherwise HELP offers you a list of topics from which to choose.

Qualifier

/REMOTE
Equivalent to the REMOTEHELP command: it accesses the remote FTP server's online help instead of the local client's online help.

Position the qualifier directly after the HELP command. If positioned after the *topic*, you could get incorrect help or an error. For example, if you specify HELP LDIR /REMOTE, you get online help for "LDIR /REMOTE," which does not exist.

LDIR

Lists files in your local directory along with their creation date and size.

See `DIRECTORY` to list files on the remote host.

See `SET DEFAULT /LOCAL` to set the default local directory.

Format

`LDIR [`*`directory`*`]`

Parameter

`directory`
Directory to list on your local host. The asterisk (`*`) wildcard is acceptable.

OPEN

Opens a connection to a remote host.

The connection remains open until you exit FTP, close the connection with the CLOSE command, or open a new connection using the OPEN command or any other command that accepts a node specification.

Format

OPEN [*host* [*username* [*password* [*account*]]]]

If you:

- Supply the host, username, password, and account (if required) with the command, you are not prompted for them separately.
- Omit the parameters from the command line, you are prompted for them.
- Use the OPEN command non-interactively (for example, a batch job), and do not want to be prompted for a username, password, and account, then include the parameters on subsequent lines, after the OPEN command, in the command file.
- Want to be prompted for a password, do not submit the command file with a batch job.

The display does not echo the password or account information. After a connection is open, you do not have to specify the parameters for remote files.

Synonym

CONNECT

Parameters

host

Name or internet address of the remote host to which you want to connect. OPEN supports any valid hostname syntax, including an internet address.

username

Username on the remote host. Enclose the username in quotes if the case is important or it contains special characters. For a null username, use a pair of quotation marks (" ").

password
Password on the remote host. Enclose the password in quotes if the case is important or it contains special characters. For a null password, use a pair of quotation marks (" ").

If you use OPEN at the DCL level (see the second example), include the password on the same command line.

Qualifiers

/PORT=*port*
Port number for the remote FTP server. If omitted, the FTP client uses port number 21.

/TIMEOUT=*time*
Timeout time, in seconds, to establish the FTP control connection. If omitted, the timeout time is 120 seconds (2 minutes). Minimum value is 20 seconds.

/TLS
Negotiate with the server to perform TLS authentication as per RFC 4217. The certificate delivered by the server is checked and self-signed certificates may be rejected if desired. After performing the negotiation user authentication takes place over an encrypted connection. Note that data transfers will not be encrypted until a PROTECTION PRIVATE command has been issued.

/VMS
/NOVMS
/VMS negotiates for VMS file structure. /NOVMS, the default, does not. If omitted, SET VMS or SET NOVMS determines the outcome (see the SET VMS command for details). Note that the OPEN /VMS and OPEN /NOVMS settings override SET VMS and SET NOVMS.

Examples

1. The following opens a connection to SYS1. If successful, you must enter a username and password.

```
FTP>OPEN SYS1
```

2. The following DCL level command opens a connection to SYS1. The line includes the username and password so that you can use the command procedure interactively or in batch processing.

```
$ FTP OPEN SYS1 "smith" "opensesame"
```

PROTECTION

Set the protection for the data port after doing TLS authentication. The `PROTECTION` command does an FTP `PBSZ` (protection buffer size) command followed by an FTP `PROT` command. The `PROTECTION` should be specified before returning the command channel to clear text mode as the RFCs specify.

Format

`PROTECTION` *`level`*

Parameters

`CLEAR`
Data transfers take place in the clear, as they would with a traditional FTP session. This is the default if no protection has been specified.

`PRIVATE`
Data transfers are encrypted such that they cannot be read by an intermediate system and are integrity protected.

Example

```
FTP>PROTECTION CLEAR
FTP>PROTECTION PRIVATE
```

PUT

Copies files to a remote host.

PUT supports full wildcard file specifications except wildcards enclosed in a quoted string. Use the /MULTIPLE qualifier for a wildcarded local-file file specification. PUT also supports use of asterisk (*) wildcards after a semicolon (;) in remote file specifications. This creates the same version in the destination file as in the source file (instead of creating a new version). If the server is not OpenVMS, the version number is part of the filename. TCPware does not issue a warning if the server host already has a higher numbered version.

Format

PUT *local-file*[,*local-file*,...] [*remote-filename*]

Synonyms and Equivalents

COPY *local-file* /LOCAL *remote-filename*
MPUT *wildcarded-local-files* [*remote-filename*] = PUT *local-file*/MULTIPLE
SEND *local-file*[,*local-file*,...] [*remote-filename*]

Parameters

local-file

Input file specification on the local host. Must conform to OpenVMS file naming rules. Use a comma between multiple file specifications.

wildcarded-local-files

Input file specification on the local host in wildcard format. Wildcards include the percent symbol (%) or the question mark symbol (?) to indicate individual characters, and the asterisk symbol (*) to indicate multiple characters. Examples of wildcarded file specifications are *.TXT, W????.*, and *.*;*.

remote-filename

Output file specification on the remote host. Enclose the file specification in quotes if you want to preserve case and did not use the SET NOLOWERCASE command. If the remote-filename is omitted, the FTP client uses the *local-file* filename and extension, unless they are part of a

quoted string. Also, enclose the file specification in quotes if it contains delimiters or symbols the FTP server can interpret in special ways.

For example, the following remote file specification is enclosed in quotes because it includes slashes (/) OpenVMS normally interprets as qualifier delimiters:

```
ALPHA"smithabcd"::"/usr/bin/proj1.txt"
```

The remote file specification must conform to the filenaming conventions of the remote host. In OpenVMS-to-OpenVMS file transfers, the *local-file* and *remote-filename* specification formats are the same. (See the *local-file* parameter).

To obtain the same version number in the destination file as in the source file (instead of creating a newer one), wildcard the destination file version using `;*`. Note that if the server is not an OpenVMS host, the version number is included in the filename. You do not get a warning if the server host already has a higher numbered version. Also, if the server host already has the version specified, the old file with that version is overwritten.

Qualifiers

If you omit one of the file type qualifiers (`/ASCII`, `/BINARY`, `/FORTRAN`, `/IMAGE`, or `/VMS`), the FTP client transfers the file based on either:

- The current default setting; for example, ASCII or IMAGE.
- The extension (type) of the file you want copied.

Setting a file type qualifier with the `PUT` command overrides the default transfer format for this `PUT` only.

See also the `SET DEFAULT` command.

/ANONYMOUS
/NOANONYMOUS
Enables (`/ANONYMOUS`) or denies (`/NOANONYMOUS`) anonymous user access to remote resources. You can omit `/ANONYMOUS` if using the file syntax *`node`*`::`*`path`*.

/APPEND
Appends the *`local-file`* file to the *`remote-filename`*. If the *`remote-filename`* file does not exist, The FTP client creates it. Some remote hosts do not support this operation.
NOTE: If the operation fails, try appending in binary mode by using the `/BINARY` qualifier.

/ASCII
Transfers the file in formatted ASCII format.

/BINARY
Transfers `.BIN`, `.LDA`, `.OBJ`, and `.STB`, files in formatted binary format.

/BLOCK
Transfers STREAM, STREAM_CR, STREAM_LF, and UNDEFINED files in block mode.

/CONFIRM
/NOCONFIRM
`/CONFIRM` issues a confirmation prompt before putting a file. Respond with `Y` or `N`. If confirming multiple file puts, use with `MPUT` or `PUT/MULTIPLE` with a wildcard value. Position the qualifier immediately after the `PUT` verb to relate to all files, or after the particular filename to relate to that file only. `/NOCONFIRM` is the default.

/CONTIGUOUS=*blocks*
Local output file should have an initial contiguous allocation of the specified number of *`blocks`*. If the output file is smaller than the specified *`blocks`*, the FTP client truncates the number of blocks. If the output file is larger, the additional allocations are non-contiguous. Does not apply to remote output files.

/CONVERT
/NOCONVERT
`/CONVERT` translates the internal file formatting characters of Variable Forms Control (VFC) files. `/NOCONVERT`, the default, does not do the conversion.

/FDL
Uses a separate FDL file describing the specified file's OpenVMS RMS record attributes. This qualifier is useful for transferring a VMS node file to a non-VMS node. A subsequent `GET /FDL` operation can then return the file with the proper record attributes back from the non-VMS node. The default is not to create an accompanying FDL file. The `TYPE` (or `SET TYPE`) command determines the type of file. A transfer of:

- ASCII data results in a sequential file with variable records (the default).
- IMAGE data results in a sequential file with fixed length records of 512 bytes.

/FORTRAN
Transfers the file in FORTRAN mode. The first character of each record is a FORTRAN carriage control character. Some hosts do not recognize this transfer format.

/IGNORE
/NOIGNORE
/IGNORE ignores errors so that copying can continue with the next file. /NOIGNORE, the default, terminates copying if an error occurs.

/IMAGE[=*size*]
Transfers the file in image mode. Optional size sets the record size of the local output file. Does not apply to remote output files.

/LOG
/LOG displays file specifications for each file transferred.

/MULTIPLE
Transfers multiple files (equivalent to MPUT). Use after *local-file* only and include wildcards in *local-file*. Necessary because some remote hosts do not recognize the OpenVMS characters for the asterisk (*), percent (%), or the question mark (?) as wildcards.

/RECORD
Transfers the file using STRU R so as to communicate the record structure during the copy. A positional qualifier. Not all servers support record structure mode. If you specify both /RECORD and /VMS, the FTP client uses /VMS.

/RESTART
For STREAM mode transfers restart the transfer where it was interrupted. The client verifies that the server supports the RFC 3659 SIZE and REST commands, and ignores the qualifier if it does not.

This does NOT work for VMS mode transfers (STRU VMS), and if the remote system is a VMS system it is recommended that a STRU FILE be done before the transfer command and to include /NOVMS on the command line.

/SET_FACTS

Set selected file facts on the destination file to match the source file after transfer. The currently supported fact is MODIFICATION__TIME.

/VARIABLE
Transfers an image file (see /IMAGE) in variable length record mode. All /IMAGE records are the same length when stored at the destination. Applies to local output image files only.

/VMS
Transfers the file in VMS file mode. Allows you to transfer any type of RMS file between OpenVMS systems. /VMS is a positional qualifier. It should immediately follow the filename in question. If you use /VMS, the FTP client ignores /APPEND, /ASCII, /BINARY, /BLOCK, /FORTRAN, /IMAGE, and /VARIABLE. If you specify both /RECORD and /VMS, the FTP client uses /VMS. Not all servers support VMS files. If the server does and you do not specify another mode (using a qualifier or the STRUCTURE or SET DEFAULT commands), /VMS is the default.

Examples

1. The following copies the STUFF.TXT file from your local host to the remote host (the receiving system stores the file under the same filename in the default directory):

```
FTP>PUT STUFF.TXT
```

2. The following copies the local STUFF.TXT file to DELTA's anonymous directory. It is equivalent to having used /ANONYMOUS:, sending the "ANONYMOUS *user-email-address*" username and password with the command.

```
FTP>SEND DELTA::STUFF.TXT
```

PWD

Prints the name of the current working directory on the remote host.

Useful for determining the default directory when not specifying a full pathname.

Format

```
PWD
```

Equivalent

```
SHOW DEFAULT
```

QUOTE

Sends an FTP command to the remote server.

> **Note:** Do not use QUOTE to initiate a file transfer operation.

Format

QUOTE *command*

Equivalents

QUOTE HELP SITE = SITE HELP = HELP /REMOTE SITE = REMOTEHELP SITE

Parameter

command
FTP command string sent to the remote FTP server. FTP commands are not the same as the FTP client commands. Enclose the command in quotes if it contains special characters, or embedded spaces, or is case-sensitive.

Example

The following sends the SYST command to the remote FTP server. If implemented by the remote server, it returns the type of operating system running on the remote server.

```
FTP>QUOTE "SYST"
```

REMOTEHELP

Accesses the remote FTP server's on-line help.

See HELP to bring up the FTP client's on-line help.

Format

```
REMOTEHELP [topic]
```

Equivalents

```
HELP /REMOTE [topic]
HELP /REMOTE SITE = REMOTEHELP SITE = SITE HELP = QUOTE HELP SITE
```

Parameter

topic

Optional topic for which you want help from the remote server. If you do not specify a topic, HELP provides you with a list of topics and prompts you to choose one.

RENAME

Renames a file on the remote host.

Format

RENAME *old-name new-name*

Parameters

old-name

File on the remote host to rename. The remote file specification must conform to the file naming conventions of the remote host. Enclose the file specification in quotes if it contains delimiters or symbols the FTP server can interpret in special ways.

new-name

Valid file specification to substitute for *old-name*. Enclose in quotes if it contains special characters, imbedded spaces, or is case sensitive.

Qualifier

/ANONYMOUS
/NOANONYMOUS

Enables (/ANONYMOUS) or denies (/NOANONYMOUS) renaming files in anonymous user directories. You can omit /ANONYMOUS if using the node file syntax (*node*::*path*).

Examples

1. The following renames the testb file to test2/test:

```
FTP>RENAME testb "test2/test"
```

2. The following renames the OLD.TXT file on DELTA to NEW.TXT. It is equivalent to using the /ANONYMOUS qualifier: sends the "ANONYMOUS *user-email-address*" username and password with the command.

```
FTP>RENAME DELTA::OLD.TXT NEW.TXT
```

SET [NO]BELL

Enables the terminal bell after completing a file transfer.

`SET NOBELL` is the default.

Format

```
SET BELL
SET NOBELL
```

Synonym

`BELL` - toggles between `SET BELL` and `SET NOBELL`

SET DEBUG /CLASS

Enables or disables displaying debugging information depending on the class keyword(s) used. The /CLASS qualifier is required.

Format

SET DEBUG /CLASS=*(keyword,...)*

Synonyms

DEBUG - toggles SET DEBUG /CLASS=COMMANDS
VERBOSE - toggles SET DEBUG /CLASS=REPLIES (default is ON)

Qualifier

/CLASS=(*keyword*,...)
Classes of debugging information to enable or disable. Use one or more of the keywords listed in the below table. The initial default is PERFORMANCE and REPLIES. Use NONE as the first entry to clear the classes before resetting them (see Example 1).

Keyword	Purpose
COMMANDS	Enables displaying FTP commands sent to the server.
PERFORMANCE	Enables displaying performance information (when using COPY/LOG, GET/LOG, or PUT/LOG).
REPLIES	Enables displaying FTP replies received from the server; equivalent to toggling the VERBOSE command ON (the default).
ALL	Enables displaying all classes.
NONE	Disables displaying all classes.

Examples

1. The following resets the debugging classes. It first disables all classes (NONE), and then enables the COMMANDS and REPLIES (VERBOSE) classes.

```
FTP>SET DEBUG/CLASS=(NONE,COMMANDS,REPLIES)
```

2. The following toggles the REPLIES (VERBOSE) class. If on, it shows informational messages (if enabled on the server) when logging in or moving around directories on the server. The ON or OFF setting is immediately displayed after the command.

```
FTP>VERBOSE
```

SET DEFAULT

Changes the default local or remote directory.

Sets the default qualifiers used with the COPY, GET, PUT, and DELETE commands.

> **Note:** Specify the parameter or the qualifiers separately. Do not specify them together.

Format

```
SET DEFAULT [directory]
```

Synonyms and Equivalents

`CD [directory]= SET DEFAULT /REMOTE` (CD allows you to use UNIX-style directory names)

```
LCD [directory]= SET DEFAULT /LOCAL
IMAGE = SET DEFAULT /IMAGE
TYPE BINARY = SET DEFAULT /BINARY
```

Parameter

directory

Default directory to set on the local or remote host, depending on whether the /LOCAL or /REMOTE qualifier follows, or the remote directory specification if no qualifier follows. The directory format is:

```
[node"username password"::]directory
```

To open a connection first, use the *node*"*username password*"**::** part of the format. This syntax is optional. The *directory* part of the format is any valid directory specification. Enclose it in quotes if it contains special characters or embedded spaces, or is case-sensitive. (You can also use the [*directory*] format, as in [-], if the remote host is an OpenVMS system.) If *directory* is omitted:

- With SET DEFAULT or SET DEFAULT /REMOTE, the FTP client sets the default directory to the parent of the current directory on the remote host.
- With SET DEFAULT /LOCAL, the FTP client sets the local default directory to your login directory defined by the SYS$LOGIN logical.

Use the *node*::*directory* syntax to access an anonymous FTP user directory, in which case you can omit the /ANONYMOUS qualifier.

Qualifiers

/LOCAL
Changes the local default directory to directory. LCD is the same as SET DEFAULT /LOCAL.

/REMOTE
Changes the remote default directory to directory. CD is the same as SET DEFAULT /REMOTE.

/ANONYMOUS
/NOANONYMOUS
Enables (/ANONYMOUS) or denies (/NOANONYMOUS) the setting of defaults for anonymous user directories. You can omit /ANONYMOUS if you use the syntax *node*::*directory*.

/[NO]APPEND
/[NO]CONFIRM
/[NO]IGNORE
/[NO]LOG
/[NO]RECORD
/[NO]VARIABLE
/[NO]VMS
These qualifiers set various transfer defaults. Do not use with /LOCAL or /REMOTE. See the COPY, GET, PUT, or DELETE command for qualifier descriptions.

/ASCII
/BINARY
/BLOCK
/FORTRAN
/IMAGE*[=n]*
These qualifiers set transfer mode defaults. Use only one. Do not use with /LOCAL or /REMOTE. See the COPY, GET, or PUT command for qualifier descriptions.

/DEFAULT
Determines the default transfer mode from the local file's file extension. Do not use with /LOCAL or /REMOTE.

Examples

1. The following equivalent commands set the local default directory to `[SMITH.DOC]`. The default device does not change.

```
FTP>SET DEFAULT /LOCAL [SMITH.DOC]
FTP>LCD [SMITH.DOC]
```

2. The following equivalent commands sets the remote default directory to `/usr/src/`:

```
FTP>SET DEFAULT /REMOTE "/usr/src/"
FTP>CD "/usr/src/"
```

3. The following sets the default transfer mode to `/IMAGE` for subsequent copy commands, and sets the default to `/LOG` and `/NOCONFIRM`:

```
FTP>SET DEFAULT /IMAGE /LOG /NOCONFIRM
```

4. The following sets the remote directory to the anonymous directory on DELTA.

```
FTP>SET DEFAULT DELTA::[]
```

It is equivalent to:

```
FTP>SET DEFAULT DELTA"ANONYMOUS user-email-address"::[]
```

5. The following sets the remote directory to `SYS$SYSDEVICE:[USER.SMITH]`:

```
FTP>CD "/sys$sysdevice/user/smith"
```

SET DEFLATE

Changes the options for `MODE DEFLATE` transfers. The only deflate engine present is ZLIB.

DEFLATE transfers can be enabled with the `SET MODE DEFLATE` command. DEFLATE transfers cannot be used when TLS is being used.

Format

```
SET DEFLATE
```

Qualifiers

/LEVEL={-1-9}
Changes the level of data compression that the engine uses when a file is transferred. The default level is `-1` - a compromise between speed and compression, `0` is no compression, `1` is best speed and `9` is best compression.

SET [NO]ALLOWSELFSIGNED

Allows or disallows self-signed certificates for RFC 4217 TLS negotiation.

The default is to allow self-signed certificates.

Format

```
SET ALLOWSELFSIGNED
SET NOALLOWSELFSIGNED
```

SET [NO]HASH

Enables hash marks.

With `SET HASH`, the FTP client displays a hash mark (#) every 1024 bytes sent or received during a file transfer. `SET NOHASH` is the default.

Hash marks appear in files only. No hash marks appear if the file transfer is output to the terminal screen.

> **Note:** With `SET HASH`, FTP reads only 1024 bytes at a time from the network layer. While this means that FTP gives more accurate reports on the progress of a transfer, it increases overhead. Use hash marks primarily with transfers over slower-speed links (such as SLIP lines).

Format

```
SET HASH
SET NOHASH
```

Synonym

`HASH` - toggles between `SET HASH` and `SET NOHASH`

SET [NO]LOWERCASE

Enables the conversion of unquoted filenames to lowercase before the FTP client sends the files to the remote host. `SET LOWERCASE` is the default.

With `SET NOLOWERCASE`, the FTP client does not convert unquoted filenames to lowercase.

The FTP client always preserves the case of filenames that appear within quotation marks.

Format

```
SET LOWERCASE
SET NOLOWERCASE
```

SET MODE

Set the transfer mode to STREAM (default), BLOCK, COMPRESSED or DEFLATE.

Format

```
SET MODE {STREAM | BLOCK | COMPRESSED | DEFLATE}
```

Parameters

STREAM
The data are transmitted as a stream of bytes. This is the default.

BLOCK
The file is transmitted as a series of data blocks preceded by one or more header bytes.

COMPRESSED
Data that contains repeated sequences may be compressed to obtain better bandwidth.

DEFLATE
The data is compressed with the ZLIB compression engine. `DEFLATE` cannot be used with sessions that use TLS authentication. The TLS code provides data compression when the data stream is protected.

Example

```
FTP>SET MODE STREAM
FTP>SET MODE DEFLATE
```

SET [NO]PASSIVE

Sets passive mode. Passive mode performs an active open on the data connection, which can avoid problems with firewall systems.

`SET NOPASSIVE` (the default) disables passive mode.

You can also define the TCPware `FTP_PASV` logical as follows:

```
$ DEFINE/PROCESS TCPWARE_FTP_PASV "TRUE"
```

Your system manager can also define the logical system-wide as follows:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_FTP_PASV "TRUE"
```

Format

```
SET PASSIVE
SET NOPASSIVE
```

Synonym

`PASSIVE` - toggles between `SET PASSIVE` and `SET NOPASSIVE`

SET [NO]VMS

Controls whether the the FTP client negotiates for VMS file structure with the FTP server when opening a connection. The default is `SET VMS`, where the client negotiates with the server to use File Descriptor Language (FDL) information.

The FTP client first queries if the server supports VMS file transfer mode. If not, it queries for VMS Plus file transfer mode, such as with the TCP/IP Services for OpenVMS (UCX) server.

In connecting to a TCPware or other OpenVMS server, the `VMS` file structure transfer mode is used.

Note that `OPEN /VMS` or `OPEN /NOVMS` overrides `SET VMS` and `SET NOVMS`.

Format

```
SET VMS
SET NOVMS
```

SHOW STATUS

Displays the following information about your present FTP session:

- Remote hostname and internet address if you are connected to a remote host
- Username on the remote host if you are connected and logged in
- Local default directory
- Remote default directory if you are logged in to a remote host and that host supports the FTP `PWD` command
- Record size to be used with the `/IMAGE` qualifier
- Defaults that are defined by the `SET DEFAULT` command for the `COPY`, `GET`, `PUT`, and `DELETE` commands

Format

`SHOW STATUS`

Synonym

`STATUS`

Example

The following shows the status for the current connection:

```
FTP>SHOW STATUS

Connected to ALPHA (192.168.1.1)
Logged in as user "SMITH"

The local default is SYS$COMMON:[SYS$LDR]
The remote working directory is /usr/users

Default qualifiers are /VMS
```

SITE

Issues a site-specific command to the remote server.

Format

SITE *command*

Equivalents

SITE HELP = HELP /REMOTE SITE = REMOTEHELP SITE = QUOTE HELP SITE

Parameter

command

Site-specific command string to send to the remote host. Enclose the command in quotes if it contains special characters or embedded spaces, or is case sensitive. Site-specific commands can vary depending on the remote FTP server; some servers do not support any.

This command is often useful in obtaining information about the site-specific commands, if any, the remote FTP server supports.

Example

The following sends a site-specific command (SITE SPAWN PRINT MYFILE.TXT) to the remote server. With the TCPware FTP server, requests printing of the MYFILE.TXT file.

```
FTP>SITE "SPAWN PRINT MYFILE.TXT"
```

SPAWN

Executes DCL commands without exiting FTP. Note that spawning is not allowed for CAPTIVE accounts.

Format

`SPAWN [command-line]`

Parameter

command-line

DCL command line you want executed. If omitted, spawns an interactive subprocess. To return from an interactive subprocess, enter `LOGOUT`.

Synonym

`Z [command-line]`

Examples

1. The following displays the time on your local host without leaving the FTP client:

```
FTP>SPAWN SHOW TIME
  3-NOV-2021 14:02:48
```

2. The following initiates DCL command mode, displays the local time, logs out, and returns to the FTP client:

```
FTP>SPAWN
$ SHOW TIME
  3-NOV-2021 14:02:51
$ LOGOUT
  Process SMITH_1 logged out at 3-NOV-2021 14:02:54.34
FTP>
```

STRUCTURE

Changes the default file structure.

The FTP client uses FILE structured files as the default. Use the `/[NO]RECORD` qualifier for the `COPY`, `GET`, or `PUT` commands to override this default for individual transactions.

Format

`STRUCTURE` *`keyword`*

Parameter

keyword

The below table lists valid values for *`keyword`*.

Value	Purpose
`FILE`	Sets FILE as the default file structure. FILE structured files consist of sequential bytes. Equivalent to `SET DEFAULT/NORECORD`. This is the default.
`RECORD`	Sets RECORD as the default file structure. RECORD structured files consists of a collection of records. Equivalent to `SET DEFAULT/RECORD`.
`VMS`	Sets VMS as the default file structure. VMS file structure allows you to transfer all types of RMS files between OpenVMS systems using File Descriptor Language (FDL) information. May OpenVMS systems that implement FTP support this structure. Equivalent to `SET DEFAULT/VMS`.

Note: Some FTP servers do not support the RECORD or VMS structures.

Example

The following changes the default file structure to FILE:

```
FTP>STRUCTURE FILE
```

TYPE

Changes the default file transfer format for all future file operations in this session.

The following rules apply to the TYPE command:

- The default file transfer format remains set until you redefine it. It does not change when opening or closing a connection.
- The default format changes only if the remote host accepts the type change.
- If there is no default file format defined, the FTP client tries to determine the file format based on the local file's file extension.

Use the COPY, GET, or PUT command qualifiers to override this default for individual transactions.

Format

TYPE *keyword*

Equivalents

```
SET DEFAULT TYPE keyword
TYPE ASCII
TYPE IMAGE
TYPE IMAGE
```

Parameter

keyword
The below table lists valid values for *keyword*.

Keyword	Purpose
ASCII	Sets formatted ASCII format. Equivalents: • SET DEFAULT/ASCII • ASCII
BINARY	Sets formatted binary format. SET DEFAULT/BINARY is equivalent.

IMAGE	Sets image format. Equivalents: • SET DEFAULT/IMAGE • BINARY • IMAGE
FORTRAN	Sets ASCII format and specifies that the first character of each record is a FORTRAN carriage control character. SET DEFAULT/FORTRAN is equivalent.
BLOCK	Sets block format. SET DEFAULT/BLOCK is equivalent.
VARIABLE	Specifies that FTP writes an image format file as a variable-length record format file. Although FTP writes the records as variable-length, all records are the same length. SET DEFAULT/IMAGE/VARIABLE is equivalent.
DEFAULT	Removes the previous default file format. SET DEFAULT/DEFAULT is equivalent. This is the default setting for an undefined format.

Examples

1. The following changes the default file format to formatted ASCII:

```
FTP> TYPE ASCII
```

2. The following removes the previous default file format. For future transactions, the FTP client tries to determine the file format based on the local file's extension.

```
FTP>TYPE DEFAULT
```

USER

Sets the username at the remote host. USER requires an open connection.

Format

USER [*username* [*password* [*account*]]]

If you:

- Supply the username, password, and account (if required) with the command, you are not prompted for them separately.
- Omit the parameters from the command line, you are prompted for them.
- Use USER in an interactive command file and do not want to be prompted for a username, enter the username in the file on the line after the USER command. (You cannot include password or account information in the interactive command file.)
- Use the command non-interactively (for example, a batch job), and do not want to be prompted for a username, password, or account, then include the parameters on subsequent lines, after the USER command, in the command file.
- Want to be prompted for a password, do not use the command file with a batch job nor specify the password in a command file.

The display does not echo the password or account information.

Synonym

LOGIN

Parameters

username
Username on the remote host. Enclose the username in quotes if case is important or if it contains special characters. Prompted if omitted.

password
Password on the remote host. Enclose the password in quotes if case is important or if it contains special characters. Prompted if omitted and required. Not echoed.

account

Account on the remote host. Enclose the account in quotes if case is important or if it contains special characters. Prompted if omitted and required. Not echoed.

Example

The following sets the username on the remote host to SMITH, and specifies a password and an account:

```
FTP>USER "SMITH" "PASSWORD" "SMITH"
```

4. Kerberos User Commands

> **CAUTION!** This chapter used to document TCPware's Kerberos V4 server, which has been deprecated and is no longer available.

5. Network Printing

The TCPware network print services include Line Printer Services (LPS) and Terminal Server Print Services. These network printing services support most printing devices, including line printers, laser printers, and plotters.

TCPware provides Internet Printing Protocol support. refer to the *TCPware Management Guide* for more information about IPP.

LPS lets users print files on printers attached to remote hosts. Users can also print files that are on a remote host to printers attached to the local host.

Terminal Server Print Services lets users print files on printers attached to terminal servers on a TCP/IP network.

TCPware bases the network printing services on:

- UNIX style LPR/LPD protocols - Line Printer Services (LPS) implement these protocols. LPS supports the UNIX style `LPR`, `LPRM`, and `LPQ` commands, and the OpenVMS style `PRINT` command. You can configure a host as an LPS client and an LPS server (LPD).

  The LPS OpenVMS print queue created during configuration can be a queue that:

  - Performs local OpenVMS print formatting and prints output on the printer associated with the remote host running LPD.
  - Sends local print requests to the remote print queue running LPD. The remote print queue performs the print formatting.
- OpenVMS print protocol - Terminal Server Print Services implements this protocol and supports the OpenVMS style PRINT command.
- Before you use the TCPware network printing services, get a list of available print queue names from your system manager and be sure that:
- TCPware print services software has been configured and started on your system.
- Any other required OpenVMS print queues have been initialized and started.

Network Print Services

Once the print queue has been initialized and started, you can send print requests to a printer attached to a remote host, or to a printer connected to a terminal server on the TCP/IP network. You can also print files that are on a remote host to printers attached to the local host.

The LPS client and Terminal Server Print Services support the following commands:

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.

The below diagram shows using the UNIX style LPR command and the OpenVMS style PRINT command when you use LPS. It also shows sending a file to a print queue associated with a terminal server on a TCP/IP network.

To send files to a printer using the networking print services:

1. Enter the LPR command to send a file to print when either the local or remote host is a UNIX system. For example:

```
$ LPR filename
```

Prints the file specified by *filename* on the default remote printer. For example:

LPR MEMO.TXT	Prints the file MEMO.TXT on the default remote printer.
LPR -PMYUNIX MEMO.TXT	Sends the file MEMO.TXT to the remote printer specified by the logical MYUNIX.
LPR -PRPRINTER1@ALPHA MEMO.TXT	Sends the file MEMO.TXT to the remote printer RPRINTER1 connected to host ALPHA. See the LPR, LPQ, LPRM, and PRINT commands in the command reference.

2. Enter the PRINT command to send a file to a print queue for printing when one of the following is true:

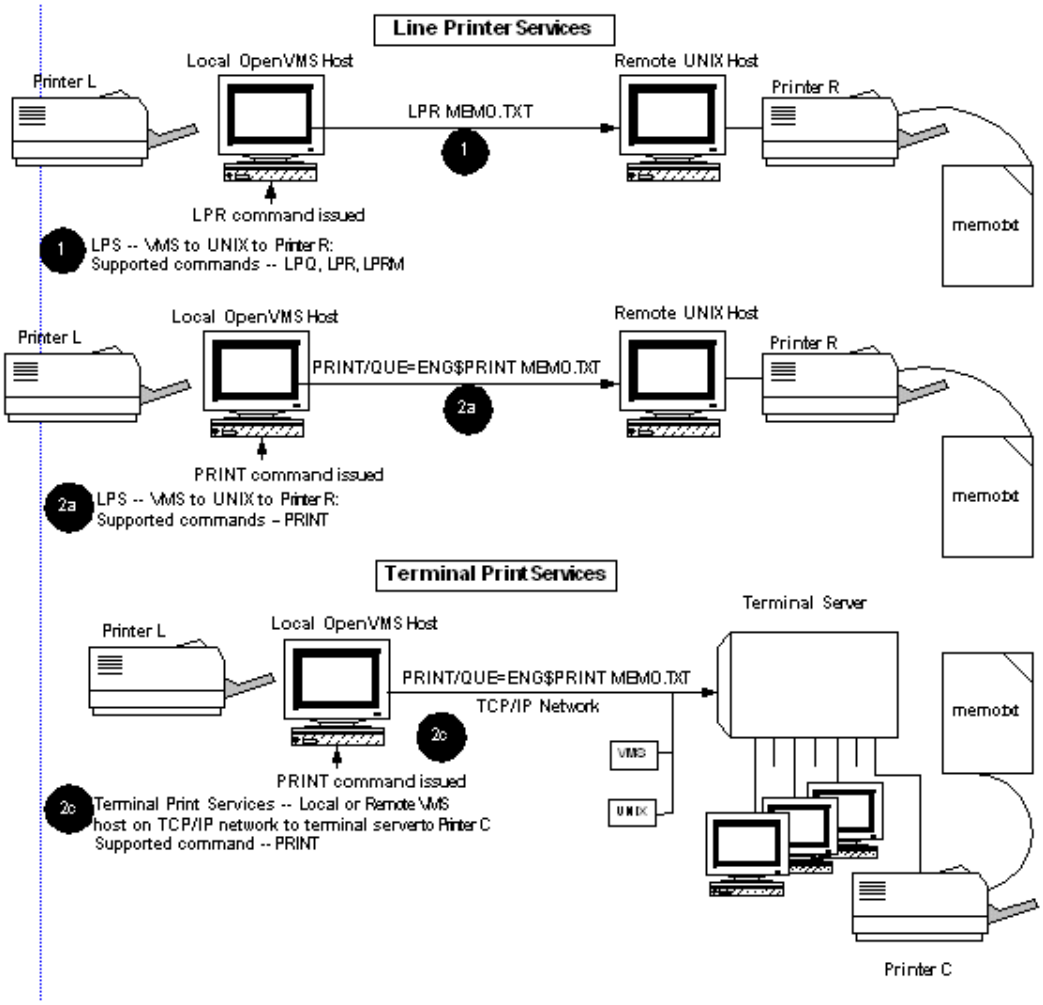
a. The local host is a TCP/IP OpenVMS host and the remote host runs the LPD server.
b. The local and remote hosts are TCP/IP OpenVMS hosts.
c. The local host is a TCP/IP OpenVMS host and the printer connects to a terminal server on a TCP/IP network.

In the print request PRINT/QUEUE=*qname filename*, the *qname* parameter is the name of the print queue, and the *filename* parameter specifies the data file or files you want printed.

For example, the print request PRINT/QUEUE=ENG$PRINT MEMO.TXT sends the file MEMO.TXT to the remote printer ENG$PRINT for printing on the printer associated with that print queue.

The standard OpenVMS qualifiers for the PRINT/QUEUE command are available.

See the *OpenVMS DCL Dictionary* for details on the PRINT command.



PRINT Qualifiers

LPS supports the OpenVMS PRINT/FORM qualifier on local LPS OpenVMS print queues. LPS OpenVMS print queues configured with the VMS formatting option support the /FORM qualifier.

LPS also supports the /PARAMETERS qualifier on remote hosts associated with the local LPS OpenVMS print queue. OpenVMS print queues configured with the LPD formatting option support the /PARAMETERS qualifier.

LPS also supports the /LIBRARY and other qualifiers associated with the OpenVMS INITIALIZE/QUEUE command. You can specify these qualifiers during CNFNET configuration.

The below diagram shows the effects of using the /FORM and /PARAMETERS qualifiers on an LPS OpenVMS queue configured for:

- VMS formatting option set up during configuration - use the /FORM qualifier
- LPD formatting option set up during configuration - use the /PARAMETER qualifier

If you intend to use the /FORM or /PARAMETER qualifier, the format of the PRINT command with the /FORM option is:

```
$ PRINT/QUEUE=qname filename /FORM=form
```

qname is the OpenVMS queue name and *form* is the form name or number.

Use the SHOW QUEUE/FORM command to display the list of the available forms for use with LPS.

The format of the PRINT command with the /PARAMETERS option is:

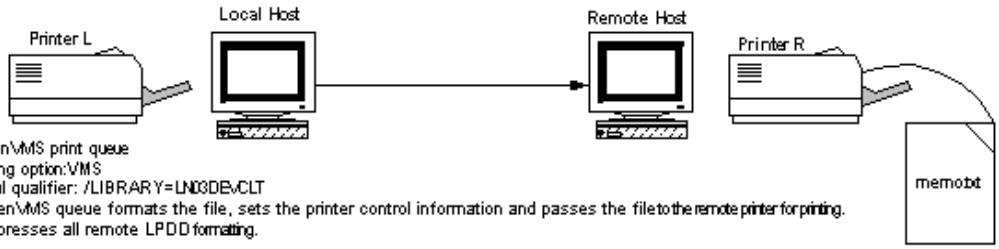
```
$ PRINT/QUEUE=qname filename /PARAMETERS=(parameters)
```

qname is the OpenVMS queue name and *parameters* is any of a number of supported parameters and their values, separated by commas, such as /PARAMETERS=(SIDES=2,NUMBER_UP=2), which indicates double-sided printing with two print "frames" (the "number up") per page.

- Ask your system manager for a list of LPS OpenVMS print queues that support these qualifiers.
- Ask your system manager for a list of available forms for LPS.

Effects of the /FORM Qualifier

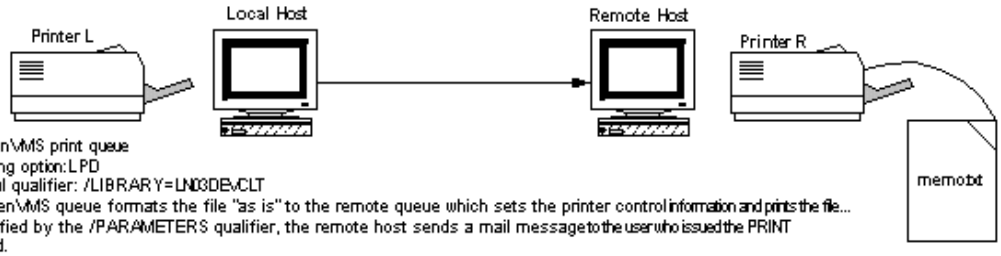
Command issued:PRINT/QUE=ENG$PRINT MEMO.TXT/FORM=FORM1



LPS Open\VMS print queue
Formatting option:VMS
Additional qualifier: /LIBRARY=LN03DEVCLT
LPS--Open\VMS queue formats the file, sets the printer control information and passes the file to the remote printer for printing.
LPS suppresses all remote LPDD formatting.

Effects of the /PARAMETERS Qualifier

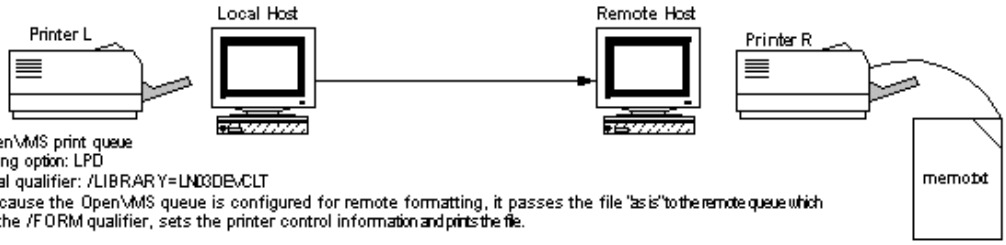
Command issued: PRINT/QUE=ENG$PRINT MEMO.TXT/PARAMETERS=("m")



LPS Open\VMS print queue
Formatting option:LPD
Additional qualifier: /LIBRARY=LN03DEVCLT
LPS--Open\VMS queue formats the file "as is" to the remote queue which sets the printer control information and prints the file...
As specified by the /PARAMETERS qualifier, the remote host sends a mail message to the user who issued the PRINT command.

Effects of the /FORM Qualifier Sent to Remote Host for Printing

Command issued: PRINT/QUE=ENG$PRINT MEMO.TXT/FORM=FORM1



LPS Open\VMS print queue
Formatting option: LPD
Additional qualifier: /LIBRARY=LN03DEVCLT
LPS--Because the Open\VMS queue is configured for remote formatting, it passes the file "as is" to the remote queue which ignores the /FORM qualifier, sets the printer control information and prints the file.

LPQ

Displays the status of specific print requests or all requests in a remote print queue.

For each request in a queue, LPQ reports the following:

- User's name
- Current rank of the request in the queue
- Names of the files within the request
- Request number
- Total size of the request in bytes

Print requests appear in the order in which you want them printed. If the filenames are unavailable (because the job consists of text entered directly from the keyboard), LPQ lists them as SYS$INPUT.

You can specify up to 50 files and 50 usernames on one LPQ command line.

```
$ LPQ [-l][-Pprinter] [job-number...] [username...]
```

You can enter commands, parameters, and options in upper or lowercase letters. Print services converts all uppercase letters to lowercase unless you enclosed them in quotation marks ("").

Parameters

job-number
Displays queue information for the specified request.

username
Displays queue information for print requests owned by a specific user.

Options

-l
Displays queue information in the long format. If you do not use this option, LPQ displays only as much information about the job as fits on one line.

-Pprinter@host
-Plogical-name

Specifies a remote print queue. If you do not use this option, LPQ displays information only for the default printer defined by the logical TCPWARE_LPR_PRINTER.

> **Note:** LPQ does not support the UNIX LPQ option +n.

Examples

1. This command displays in short form all jobs queued to the printer sys$print on host daisy and owned by user smith:

```
$ LPQ
lp is ready and printing
Rank     Owner      Job    Files                 Total Size
active   smith      45     memo1, memo2, memo    3957 bytes
1st      jones      46     prog.c                 897 bytes
2nd      ross       47     letter.txt             432 bytes
```

2. This command displays in long form all jobs queued to the printer sys$print on host daisy and owned by user smith:

```
$ LPQ -l -Psys$print@daisy smith
lp is ready and printing
smith: active            [job 045daisy.example.com]
   3 copies of memo.txt             957 bytes

smith:1st                [job 046daisy.example.com]
   prog.c                           897 bytes

smith:3rd                [job 048daisy.example.com]
   letter.txt                       432 bytes
```

3. This command displays job 489 in the queue for the default remote printer:

```
$ LPQ 489
lp is ready and printing
Rank    Owner     Job     Files                 Total Size
active  gordon    489     aug.txt, sept.txt     560 bytes
```

LPR

Sends a file to a remote print queue.

If you omit a file specification, the job consists of data you type from the keyboard.

The TCPWARE_LPR_PRINTER logical defines the default remote printer.

TCPware creates the LPR temporary file in SYS$SCRATCH. In this way, if you have a limited disk quota, you can print by redefining the SYS$SCRATCH logical to point to a public scratch disk that has no disk quota limitations.

Format

LPR [*option*...] [*filespec* ...]

Parameter

filespec
Name of the file(s) you want queued. Use the asterisk (*) or percent sign (%) as a wildcard character. Enclose in quotes if you want to preserve case other than all lowercase. For multiple files, leave a space between each *filespec*. The default extension is .LIS.

Options

The following options are listed in the order characters (lowercase and uppercase), numbers, and symbols. They are all prefixed by a hyphen and some take arguments. The lowercase and uppercase character options can mean very different things and are listed together for comparison sake. The important distinction is that the uppercase options all take arguments. There are two ways to keep this distinction clear on the command line:

Enter lowercase options in lowercase and uppercase options in uppercase	Here you MUST enclose the uppercase character in quotes; for example, -"P" ("use the remote printer indicated by the following argument"). Also include a space character between a lowercase (unquoted) option and any *filespec*, or the entry will be interpreted as an option that takes an argument (see the next method).

Enter all options in lowercase	Here you MUST distinguish the options taking arguments by appending the argument *immediately* after the option character (with no intervening space). For example, `-plp` means "use remote printer `lp`", while `-p lp` (with the space) means "print the `lp`*`file`*, which contains UNIX `pr` formatting commands."

`-c`
File contains data in the UNIX CIF graphics language.

`-"C"`*`job-classification`*` (or -c`*`job-classification`*`)`
Names the job classification you want used on the burst page. If you omit this option, the job classification is the domain name of the local host. (See the previous note for details on syntax.)

`-d`
File contains output from TeX formatting commands.

`-f`
Uses a filter that interprets the first character of each line in the file as a standard FORTRAN carriage control character.

`-g`
File contains standard UNIX plot data as produced by the plot routines.

`-h`
Suppresses the printing of the burst page.

`-i[`*`number`*`]`
Indents the output the specified number of blank spaces. If you do not enter a *`number`*, the output indents eight spaces. (Do not leave a space between the `-i` and the *`number`*.)

`-"J"`*`job`*` (or -j`*`job`*`)`
Prints the job name on the burst page. If you do not use this option, the job name is the name and extension of the first file in the job.

`-l`

Uses a transparent filter so that you can send data to the printer unchanged. Note that the data is UNCONVERTED; print services does not convert the files to STREAM-LF format. Use this option with BINARY data or files containing all of the characters, including carriage returns (CRs), when you want them sent to the printer.

-m
Sends a mail message to the user who issued the LPR command upon completion of the job. You can use this option only if your local host implements the Simple Mail Transfer Protocol (SMTP). This also sets the /NOTIFY option for PRINT, so that if you are logged in as the user under which the job was printed, you will be notified that the job completed.

-n
File contains UNIX ditroff (device independent troff) formatting commands.

-o
File contains PostScript input.

-p
Prints the file with page headers. (Do not append any characters onto the p of the option or it can be interpreted as an argument to the uppercase -P option.)

-"P"*printer*@*host* (or -p*printer*@*host*)
-"P"*logical* (or -p*logical*)
Specifies a remote printer. If you do not use this option, lpr uses the default printer defined by the logical TCPWARE_LPR_PRINTER.

-r
Deletes the files from your local host after sending them to the remote queue. Use this option cautiously. The remote host deletes the file when accepting the job. However, the remote host does not guarantee that it will print or execute the job. (That is, the remote printer might fail, someone could delete jobs from the queue, or you might not have access to the queue). The remote host does not delete the file if the remote queue does not accept the job.

-t
File contains output from UNIX troff formatting commands. (Do not append any characters onto the t of the option or it can be interpreted as an argument to the uppercase -T option.)

-"T"*title* (or -t*title*)
Prints a title on the first page of output. Use this option only when you use the `-p` option to format a file.

-v
File is in Sun raster format.

-w*number*
Width of the output pages in characters. (Do not leave a space between the `-w` and the *number*.)

-z*number*
Length of the output pages in lines. (Do not leave a space between the `-z` and the *number*.)

-1*string*
-2*string*
-3*string*
-4*string*
The options name UNIX font files and work the same as they do in UNIX. (Do not leave a space between the number and the string.)

Use these options only with the `-d`, `-n`, and `-t` options.

-#*number*
Prints multiple copies, where *number* is the number of copies you want of each file. (Do not leave a space between the `-#` and the *number*.)

> **Note:** LPR does not support the UNIX `lpr` options `-s` and `-q`. Some LPD servers that reside on non-UNIX hosts (such as the one provided by TCPware) do not support the following UNIX `lpr` options: `-p`, `-t`, `-n`, `-d`, `-g`, `-v`, `-c`, `-i`, `-w`, `-z`, `-1`, `-2`, `-3`, and `-4`.

Examples

1. This command prints the file `MEMO.TXT` on the default remote printer:

```
$ LPR MEMO.TXT
```

2. Each of these commands send the file MEMO.TXT to the remote printer specified by the logical drp02:

```
$ LPR -"P"drp02 MEMO.TXT
$ lpr -pdrp02 memo.txt
```

3. Each of these commands send the file MEMO.TXT to the remote printer lp at host daisy:

```
$ LPR -"P"lp@daisy MEMO.TXT
$ lpr -plp@daisy memo.txt
```

4. Each of these commands specify mymemos as the job name on the burst page, and send MEMO1.TXT, MEMO2.TXT, and MEMO3.TXT to the default remote printer:

```
$ LPR -"J"mymemos MEMO1.TXT MEMO2.TXT MEMO3.TXT
$ lpr -jmymemos memo1.txt memo1.txt memo2.txt memo3.txt
```

5. This command sends three copies of the MEMO.TXT and LETTER.TXT files to the default remote printer:

```
$ LPR -#3 MEMO.TXT LETTER.TXT
```

6. This command indicates that the file contains UNIX troff formatting code, the burst page should not be printed, and the width of the output should be 72 characters.

```
$ LPR -t -h -w72 MEMO.LIS
```

LPRM

Removes one or more jobs from a remote print queue. You can remove jobs from remote queues in these situations only:

- The jobs were submitted from your local host
- Your local host has direct access to the remote host. The following files define this access:
  - `TCPWARE:LPD_USERS.DAT` (the LPD access File) for TCPware hosts
  - `etc/hosts.lpd` or `/etc/hosts.equiv` for UNIX hosts

When removing remote jobs from an OpenVMS host, use the `LPRM` command instead of the OpenVMS `DELETE/ENTRY` command. `LPRM` removes files from the `TCPWARE_LPD_SPOOL` directory, whereas `DELETE/ENTRY` does not.

The `LPRM` command displays a message only when it removes a job or encounters an error. If it does not delete a job (such as when the queue is empty), a message does not appear.

You can specify up to 50 jobs and 50 usernames on one `LPRM` command line.

Format

`LPRM [-"P"`*`printer`*`] [`*`job-number ...`*`] [`*`username ...`*`] ["-"]`

TCPware converts all uppercase letters to lowercase unless you enclose them in quotation marks (`""`).

If you omit a *`job-number`* or *`username`* and you own the job that is currently active, TCPware removes the job.

Parameters

job-number
Specifies which job you want removed from the remote queue. If you omit this parameter, TCPware removes the currently active job.

Use the `LPQ` command to display the *`job-number`* of a job.

username
Specifies the owner of the jobs you want removed from the remote queue. TCPware removes all jobs the specified user owns.

You can remove jobs that you do not own from a remote queue only under these conditions:

- The remote host is an OpenVMS host
- Your local account is mapped to an OpenVMS username that has OPER privilege on the remote host

Use the LPQ command to display the *usernames* for all jobs.

Options

-"P"*printer@host*
-"P"*logical-name*
Specifies a remote printer. If you omit this option, TCPware removes the job from the queue the `TCPWARE_LPR_PRINTER` logical defines.

"-"
If you have OpenVMS OPER privileges on the local host, this option removes all jobs your local host submitted to the remote queue. Otherwise, it removes only your jobs.

Place quotation marks (`" "`) around this option if it is the last character on the command line because OpenVMS treats trailing hyphens as continuation line indicators.

Do not enter this option when you enter the *`job-number`* or *`username`* parameters.

Examples

1. This command removes your currently active job from the default remote print queue:

```
$ LPRM
```

2. This command removes all jobs that belong to user smith from the lp queue on host daisy:

```
$ LPRM -"P"lp@daisy SMITH
```

3. This command removes jobs 489, 490, and 495 from the default remote print queue. You can issue this command if you own these jobs, or you have OpenVMS OPER privilege on the remote host:

```
$ LPRM 489 490 495
```

4. If you have OpenVMS OPER privilege on the local host, this command removes all jobs from the default remote print queue. If you do not have this privilege, this command removes only the jobs you own.

```
$ LPRM "-"
```

PRINT

Queues jobs for printing on a local or remote printer. Useful for sending a print job to a printer attached to a terminal server.

For details on Terminal Server Print Services implementation, see the /QUEUE qualifier.

The OpenVMS process that controls OpenVMS queues determines the remote printer by checking the following items in this order:

1. The TCPWARE_LPR_*qname*_PRINTER system logical
2. The /PARAMETERS qualifier
3. The TCPWARE_LPR_*qname*_PRINTER_DEFAULT system logical

Information in this section applies only to using the TCPware for OpenVMS PRINT command with LPS and Terminal Server Print Services.

The OpenVMS documentation provides complete information on the PRINT command.

Format

PRINT *file-spec*[, *file-spec*...]

Parameter

file-spec
Specifies the file (or files if separated by commas) you want printed.

Qualifiers

/COPIES=*n*
Prints multiple copies of output, where n is the number of copies.

If you place this qualifier immediately after the PRINT command, each file listed in the command string prints *n* times. (The same effect occurs when you use the -#*number* option with the LPR command.) If you place this qualifier after a file specification, only that file prints *n* times.

/FORM=*form-name*

Specifies the name or number of the form you want associated with the print job. If omitted, the default form for the execution queues with the job.

Forms have attributes such as print image width and length or paper stock. To see which forms are defined for your system, use the SHOW QUEUE/FULL command.

/NAME=*job-name*
Names the job. If you do not use this qualifier, the job name is the name and extension of the first file in the job. This name displays on the screen when you use the LPQ command to request queue information, and on the flag page.

This qualifier is equivalent to the -J option used with the LPR command.

/NOFLAG
Suppresses printing of the burst page.

This qualifier is equivalent to the -h option used with the LPR command.

/NOTE=*string*
Names the job classification you want used on the burst page. If you omit this qualifier, the job classification is the domain name of the local host.

/PARAMETERS=(*parameter-1*[,...,*parameter-8*])
Allows you to specify UNIX lpr command options that do not have OpenVMS equivalents. If you enter only *parameter-1*, you can omit the parentheses. You can enter up to eight parameters:

parameter-1	sends jobs to a specific remote printer. Enter either a system logical name or *printer*@*host*. This parameter overrides the printer defined by the TCPWARE_LPR_*qname*_PRINTER_DEFAULT logical. If you choose to use the default printer and want to enter subsequent parameters in the same command line, you must enter double quotation marks ("") in place of *parameter-1*.
parameter-2 through *parameter-8*	specify the following LPR UNIX options: c, d, g, i, l, m, n, o, p, t, T, v, x, w, z, 1, 2, 3, 4. You can use leading hyphens, but they are not required. Enclose the option in quotation marks (for example, "t"). (The f option is unnecessary; the OpenVMS process that controls OpenVMS print queues automatically specifies this filter for FORTRAN carriage control files.)

Each parameter can include more than one option. However, you must enclose all options within the same set of quotation marks (for example, "m g", "i d")

> **Note:** LPR does not support the UNIX lpr options -s and -q. Some LPD servers that reside on non-UNIX hosts (such as the one provided by TCPware) do not support the following UNIX lpr options: -p, -t, -n, -d, -g, -v, -c, -i, -w, -z, -1, -2, -3, and -4.

/PASSALL/NOPAGE
Uses a transparent filter so that you can send data to the printer unchanged. Note that this command qualifier DOES NOT convert the file to STREAM-LF format. This qualifier is equivalent to the -l option used with the LPR command.

When using *-to-OpenVMS printing, the /PASSALL qualifier prints text files without carriage returns (CRs). Use this option mainly with BINARY data or a file that contains all of the characters (including CRs) that you want sent to the printer.

If you use LPS and issue the PRINT command, the printing process ignores the /BURST, /CHARACTERISTIC, /HEADER, /PAGES, /SETUP, /SPACE, and /TRAILER OpenVMS PRINT qualifiers. All other OpenVMS PRINT qualifiers work the same as they normally do with OpenVMS.

/QUEUE=*qname*
Specifies a print queue that can send the job to a local or remote printer. If you omit this parameter with Line Printer Services, the job goes to the SYS$PRINT queue.

The /QUEUE parameter is necessary when generating a print request on a remote printer attached to a terminal server (when using the Terminal Server Print Services). Once the server initializes and starts the print queue for a terminal server print job, you can generate a print request on the terminal printer as follows:

```
$ PRINT/QUEUE=qname filename
```

The *qname* parameter is the name of the print queue, and the *filename* parameter specifies the data file or files you want used. The standard OpenVMS qualifiers are available.

VMSLPR Symbiont

By default the VMSLPR symbiont generates a flag page locally using the VMS print symbiont and suppresses the banner page generated by the LPD server. You can make the VMSLPR symbiont request a banner page from the LPD server on a specific queue by defining the logical:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_VMSLPRSMB_queue_REMOTE_BANNER "TRUE"
```

To enable this functionality on all VMSLPR symbionts, define the logical:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_VMSLPRSMB_REMOTE_BANNER "TRUE"
```

The following logical has been added to the VMSLPR symbiont allowing you to define the number of characters you want removed from the end of a print job.

```
$ DEFINE/SYS/EXEC TCPWARE_VMSLPRSMB_queue_TRIMTAIL #
```

> **Note:** # is a numeric value indicating the number of characters to remove from the end of each print job. If not specified, the default value is 2.

Examples

1. This command prints the file `MEMO.TXT` on the remote default printer:

```
$ PRINT/QUEUE=LPR$PRINT MEMO.TXT
```

2. This command sends the file `MEMO.TXT` to the `SYS$PRINT` queue, which is usually a local printer:

```
$ PRINT MEMO.TXT
```

3. This command prints the file `MEMO.TXT` on the `lp` printer at host DAISY. You can enter this command only if you did not define the system logical `TCPWARE_LPR_LPR$PRINT_PRINTER`.

```
$ PRINT/QUEUE=LPR$PRINT /PARAMETERS="lp@DAISY" MEMO.TXT
```

4. This command is identical to the previous example, with these additions:

- The user who issued the command receives a mail message when the job completes.
- The file contains UNIX troff commands.

```
$ PRINT/QUEUE=LPR$PRINT /PARAMETERS=("lp@DAISY","m","t") MEMO.TXT
```

5. This command is identical to the previous example except that *parameter-1* is omitted:

```
$ PRINT/QUEUE=LPR$PRINT /PARAMETERS=("","m","t") MEMO.TXT
```

The result is that the file MEMO.TXT goes to the printer defined by the TCPWARE_LPR_LPR$PRINT_PRINTER_DEFAULT logical.

PRINT Command Options

Print command options are specified using the OpenVMS standard `/PARAMETERS` qualifier. The list of options is enclosed in parenthesis. For example,

```
$ PRINT /QUEUE=IPP_PRINTER_1 -
$_ /PARAMETER=(COPIES=3, ORIENTATION=LANDSCAPE) FILE.TXT
```

These options are not case sensitive. The underscores in the option names are optional. Each may be abbreviated as long as the result is not ambiguous.

The available print command options are:

PRINTER=*printer_uri*
Specifies the target printer when the queue default is not desired, or when there is no queue default. The printer URI specified must match at least one of the defined `printer_uri`'s for the print queue.

Wildcards cannot be used in the printer URI.

COPIES=*number*
Specifies the number of copies of each document to print. The default value is 1.

SIDES=*keyword*
Specifies how the printing is to be placed on the paper. The *`keyword`* must be one of the following:

- `ONE-SIDED` or `1sided`: prints each consecutive page upon one side of consecutive media sheets.
- `TWO-SIDED-LONG-EDGE` or `two-long-edge` or `2long_side`: prints each consecutive pair of pages upon the front and back sides of consecutive media sheets, with the orientation of each pair of pages on the long edge. This positioning is called “duplex” or “head-to-head” also.
- `TWO-SIDED-SHORT-EDGE` or `two-short-edge` or `2short_side`: prints each consecutive pair of pages upon front and back sides of consecutive media sheets, with the orientation of each pair of print-stream pages on the short edge. This positioning is called “tumble” or “head-to-toe” also.

ORIENTATION=*keyword*
Specifies the page orientation. The *`keyword`* must be one of:

- `PORTRAIT`
- `REVERSE_PORTRAIT`
- `LANDSCAPE`

- REVERSE_LANDSCAPE

These can be abbreviated to any non-ambiguous prefix. Case is ignored.

[NO]FLAG
Requests, or suppresses, the printing of an IPP flag page for the job. The printer may, or may not, respond to this request. The exact format of this flag page is up to the IPP Server (printer) implementation.

NUMBER_UP=*number*
Specifies the number of page images to be placed on each side of each sheet of paper. The number must be an integer that is acceptable to the IPP server. If the number specified is not a value supported by the server, the job aborts.

DOCUMENT_FORMAT=*MIME-media-type* or
DOCUMENT_FORMAT=*printer_default*****
Specifies the document format of the files in the job, or specifies use of the printer's built-in default. The default for this qualifier is the default for the queue. Also, if the queue configuration does not specify a default document format, the hard coded default is `text/plain`.

JOB_PRIORITY=*integer*
Specifies the priority of the print job at the IPP server (not to be confused with the OpenVMS queue priority). 1 is the lowest, 100 is the highest.

FINISHINGS="*keyword[,keyword]…*"
Specifies finishing operations to be performed on the printed documents. May or may not be supported by a given IPP server. Any or all of the four available finishings may be specified. Case is ignored.

- BIND
- COVER
- PUNCH
- STAPLE

MULTIPLE_DOCUMENT_HANDLING=*keyword*
Specifies how you want the printer to print your job. The *keyword* is one of the following:

- Single_Document or 1Document
- Separate_Documents_Uncollated_Copies or UncollatedSeparate
- Separate_Documents_Collated_Copies or CollatedSeparate

- Single_Document_New_Sheet or NewSheet

Case is ignored. See /MULTIPLE_DOCUMENT_HANDLING_DEFAULT in Chapter 15 of the *TCPware Management Guide* for information on single document, separate-documents-uncollated-copies, separate-documents-collated-copies, and single-document-new-sheet handling.

PAGE_RANGES="*range[,range]...*"
Specifies the page numbers to print. *range* is either a single integer page number, or a pair of page numbers, separated by a hyphen. Multiple range specifications are separated by commas and enclosed in double quotes.

For example:

```
$ PRINT/QUEUE=IPP_QUEUE/PARAM=(PAGE_RANGES="1,3-6,9,10,12-14")
FILE.TXT
```

Note that embedded spaces are allowed, and ignored. The example specifies the pages: 1, 3, 4, 5, 6, 9, 10, 12, 13, and 14.

MEDIA=*name*
This attribute identifies the medium that the printer uses for all pages of the job.

The values for *media* include medium names, medium sizes, input trays and electronic forms. See your printer documentation for details concerning what values are supported for your printer.

Standard keyword values are taken from ISO DPA and the Printer MIB and are listed in section 14 of RFC 2566. Some servers may support definition of locally created names as well.

See the below tables for the standard media names.

QUALITY=*keyword*
Specifies the quality of the printed material. Case is ignored. The keyword choices are:

- DRAFT
- HIGH
- NORMAL

The below table contains examples of standard names. These names include, but are not limited to the following:

Name	Description
default	The default medium for the output device
iso-a4-white	Specifies the ISO A4 white medium
iso-a4-colored	Specifies the ISO A4 colored medium
iso-a4-transparent	Specifies the ISO A4 transparent medium
na-letter-white	Specifies the North American letter white medium
na-letter-colored	Specifies the North American letter colored medium
na-letter-transparent	Specifies the North American letter transparent medium
na-legal-white	Specifies the North American legal white medium
na-legal-colored	Specifies the North American legal colored medium
na-9x12-envelope	Specifies the North American 9x12 envelope medium
monarch-envelope	Specifies the Monarch envelope
na-number-10-envelope	Specifies the North American number 10 business envelope medium
na-7x9-envelope	Specifies the North American 7x9 inch envelope
na-9x11-envelope	Specifies the North American 9x11 inch envelope
na-10x14-envelope	Specifies the North American 10x14 inch envelope
na-number-9-envelope	Specifies the North American number 9 business envelope
na-6x9-envelope	Specifies the North American 6x9 inch envelope
na-10x15-envelope	Specifies the North American 10x15 inch envelope

executive-white	Specifies the white executive medium
folio-white	Specifies the folio white medium
invoice-white	Specifies the white invoice medium
ledger-white	Specifies the white ledger medium
quarto-white	Specified the white quarto medium
iso-a0-white	Specifies the ISO A0 white medium
iso-a1-white	Specifies the ISO A1 white medium
a	Specifies the engineering A size medium
b	Specifies the engineering B size medium
c	Specifies the engineering C size medium
d	Specifies the engineering D size medium
e	Specifies the engineering E size medium

The following standard values are defined for input trays:

Name	**Description**
top	The top input tray in the printer.
middle	The middle input tray in the printer.
bottom	The bottom input tray in the printer.
envelope	The envelope input tray in the printer.

manual	The manual feed input tray in the printer.
large-capacity	The large capacity input tray in the printer.
main	The main input tray
side	The side input tray

Submitting Jobs to IPP Symbiont Print Queues

This section describes how to submit jobs to the IPP symbiont print queues.

Printing a Single Text File to an IPP Queue

Print the file FOO.TXT to the IPRINTER (default destination printer) set up in the prior examples:

```
$ PRINT/QUEUE=IPRINTER_QUEUE foo.txt
```

Specifying the Destination Printer on the Print Command

Print a single text file to a non-default printer on a queue with a wild carded printer URL:

```
$ PRINT /QUEUE=iprinter_queue -
_$ /PARAM=(printer="ipp://another.mynet.com/ipp/port1") foo.txt
```

> **Note:** The above will fail unless the queue specifies another.mynet.com as a legal URL, either explicitly or by using wildcards.

Using Other Print Qualifiers

Print a text file to a default printer on a queue but specify the document format and additional copies:

```
$ PRINT /QUEUE=iprinter_queue-_$
_$ /PARAM=(document="text/plain" copies=3) foo.txt
```

TCPware IPP SHOW Command

The TCPware `IPP SHOW` utility allows a user to learn the capabilities supported by an IPP server. This utility queries the server and displays the supported attributes. The program can be used to check on the capabilities of a given server. When called from a DCL script or other program, it can be used to gather information about a number of printers or used to match printer capabilities with the needs of a given print job.

For detailed information on the `IPP SHOW` command, see Chapter 15 of the *TCPware Management Guide*.

6. RCD and RMT: Remote CD-ROMs and Tapes

Introduction

The Remote Magnetic Tape (RMT) client and Remote Compact Disc (RCD) client provide access to tape drives and CD-ROM drives, respectively, on remote TCP/IP systems. This chapter describes how to set up RMT and RCD on your OpenVMS system so that you can use the commands typically associated with tape and CD-ROM drives, such as `BACKUP`, `MOUNT`, `COPY`, and `EXCHANGE`.

RMT Client and RCD Client

To use a remote tape or CD-ROM, you must first "connect" to the server system with the `RMTSETUP` command, which creates a pseudo device. You can then use OpenVMS commands such as `BACKUP`, `MOUNT`, `COPY`, and `EXCHANGE`. These are the same commands issued directly to the physical tape or CD-ROM device on the server. When you conclude the activity, you can discard the pseudo device using the `DEALLOCATE` command.

Note that not all tape drives or CD-ROM drives can fully support use of the RMT client and RCD client. For example, quarter-inch tape drives on UNIX systems typically support only fixed-length, 512-byte records. You cannot use these tapes with the OpenVMS `COPY` or `BACKUP` commands because the latter require variable-length records.

An attempt to perform an unsupported operation to a remote device results in a `%SYSTEM-E-UNSUPPORTED` error message.

Troubleshooting

You can lose the TCP/IP connection between the RMT or RCD client and server if:

- An RMT or RCD server receives a command that it does not recognize. Rather than returning an error message, it simply closes the connection.
- A bug in an RMT or RCD server causes it to crash.
- If the RMT or RCD server (or its system) crashes or is shut down.

In these situations, the RMT client or RCD client detects the loss of the TCP/IP connection and returns the following error message for all subsequent commands:

```
%SYSTEM-F-LINKABORT, network partner aborted logical link
```

The only alternative at this point is to deallocate the existing device and reconnect to the server (when it becomes available) by running `RMTSETUP` again.

RMTSETUP

Configures an RMT or RCD pseudo device, _RMT*n*: or _RCD*n*:, respectively, on your local OpenVMS system. In this way you can perform functions on remote magnetic tape or CD-ROM drives connected to an RMT or RCD server. The remote RMT or RCD server must support the rmt protocol.

Connecting to a remote CD-ROM drive requires the /CD qualifier. You can connect to the remote host with a different username by specifying the optional /USERNAME qualifier on the command line.

Format

RMTSETUP *host remote-device* [*logical*]

Parameters

host

Name or internet address of the host on which the remote tape or CD-ROM drive resides. This host must have an RMT server available.

remote-device

Name of the remote tape device (such as MKB500:) or CD-ROM device (such as DKA200:) on the RMT server. If sending the device and any server options to a non-TCPware server, you must enclose this information in double quotes, such as "/dev/rst0" for a UNIX server with "read-only" privileges.

logical

Optional OpenVMS logical assigned to the newly created pseudo device. If omitted, RMTSETUP uses the logical name TCPWARE_TAPE for tapes and TCPWARE_DISK for disks.

Qualifiers

Not all RMT servers support the following RMT Client qualifiers as options or qualifiers. For UNIX servers, for example, you must include options as part of *remote-device* as a quoted string. For example, "/dev/mt0" is a stream device and "/dev/rmt0" is a non-stream device. With a TCPware RMT server, where *remote-device* is not a quoted string, the client qualifiers that are also server qualifiers are sent to the server.

/ASSIST
/NOASSIST
Action to take when the device cannot mount on the remote system. With /ASSIST (the default), operator messages appear on the remote system indicating corrective action to take (if supported). With /NOASSIST, only a local message appears. Not allowed when used with /CD.

/BLOCKSIZE=*size*
Default block size of the remote tape device. Not allowed when used with /CD.

/CD
Indicates that the remote device is a CD-ROM device.

/COMMENT="*string*"
Used with the /ASSIST qualifier to send a message to the remote operator when a mount operation fails. Not allowed when used with /CD.

/DENSITY=*density*
Density, in bits per inch, at which to write the remote tape. Not allowed when used with /CD.

/LOG
/NOLOG
/LOG displays log information during RMTSETUP execution. /NOLOG, the default, does not.

/MOUNT
/NOMOUNT
/MOUNT, the default, allows the user exclusive access to the device. /NOMOUNT disables exclusive access to the device. /NOMOUNT also prevents a remote tape from rewinding when deallocating the pseudo device on the client. Not allowed when used with /CD.

You cannot combine /NOMOUNT with /ASSIST, /BLOCKSIZE, /COMMENT, or /DENSITY.

Use /NOMOUNT carefully since it allows multiple users access to the same device.

/PASSWORD*[=password]*
/NOPASSWORD
/PASSWORD sets the password to access the remote system and causes the RMT server to use the rexec rather than the rshell service. The *password* is converted to lowercase unless you enclose

it in quotes. /NOPASSWORD uses the rexec service with a blank password. Without either qualifier, access to the remote tape device is controlled through the TCPWARE:HOST.EQUIV and SYS$LOGIN:.RHOSTS files. Use together with /USERNAME.

Using the *password* value can pose a security risk. Also, using a null password for which you have to be prompted can cause an error in a command procedure.

/REWIND
/NOREWIND
/REWIND, on by default, rewinds a tape before its initial use. /NOREWIND causes the tape to stay in an arbitrary position after running RMTSETUP. Not allowed when used with /CD.

/STREAM
/NOSTREAM
A tape is normally written as a series of records. /STREAM ignores record boundaries and returns data read from the tape as a stream of bytes (the UNIX model). Not allowed when used with /CD.

Most OpenVMS utilities expect tape drives to operate in non-stream mode, so take care in overriding the /NOSTREAM default.

/TRUNCATE_USERNAME*[=length]*
Truncates the username sent to the RMT server to the specified *length* to accommodate requirements of some non-OpenVMS systems. The default *length* is 8.

/UNLOAD
/NOUNLOAD
/UNLOAD, the default, unloads the remote device when deallocating the local RMT pseudo device. /NOUNLOAD disables this. Note that a DCL MOUNT or DISMOUNT with /UNLOAD or /NOUNLOAD overrides the RMTSETUP /UNLOAD or /NOUNLOAD. Not allowed when used with /CD.

/USERNAME=*username*
Username for access to the remote system. If omitted, the username of the client process is sent to the server (subject to truncation by /TRUNCATE_USERNAME). *username* is converted to lowercase unless you enclose it in quotes. Use together with /PASSWORD.

/WRITE
/NOWRITE

Writing to the remote tape is usually enabled with the default /WRITE. /NOWRITE is a precautionary measure to prevent a remote tape from being written. Not allowed when used with /CD.

Examples

1. This example uses tape drive MKB500: on remote OpenVMS system IRIS to back up all the TCPware data files that start with SM. The tape is left loaded in the drive after its use (/NOUNLOAD). MYTAPE is the logical name for the _RMT9: device created.

```
$ RMTSETUP IRIS MKB500: MYTAPE /NOUNLOAD /LOG
Connecting to RMT server on host IRIS through port 514 (rsh)
Opening MKB500:/NOSTREAM/NOUNLOAD
_RMT9: created
$ BACKUP /LOG TCPWARE:SM*.DAT MYTAPE:TCPWARE.BCK /SAVE_SET
%MOUNT-I-MOUNTED, TEST1 mounted on _RMT9:
%BACKUP-S-COPIED, copied SYS$SPECIFIC:[TCPWARE]SM.DAT;1
%BACKUP-S-COPIED, copied SYS$SPECIFIC:[TCPWARE]SM_BAK.DAT;1
$ DISMOUNT MYTAPE /NOUNLOAD
$ DEALLOCATE MYTAPE
```

2. This example requests access to tape drive /dev/rst0 on a remote UNIX system, using a username and password. The initialize command was unrecognized by the tape drive on the UNIX system and rejected. The tar utility examines the contents of the tape, which was written from the UNIX system. (tar is available over the network and is an alternative to the EXCHANGE utility).

```
$ rmtsetup sigma.nene.com "/dev/rst0" -
_$ /username=system /password
Password for root on host SIGMA.EXAMPLE.COM: ******
$ initialize tcpware_tape test
%INIT-F-UNSUPPORTED, unsupported operation or function
$ mount /foreign /record_size=512 tcpware_tape
$ tar -ftv tcpware_tape
644   4069 Jun 1 16:29:21 2021 /etc/hosts
End of Tar file found.
Do you wish to move past the EOF mark (y/n)? n
$ dismount tcpware_tape
$ deallocate tcpware_tape
```

3. This example requests access to CD-ROM drive DKA100: on remote host roman, mounts the CD-ROM using MY_CD as the logical name, and requests a directory listing:

```
$ rmtsetup /cd /log roman dka100: my_cd
Connecting to RCD server on host ROMAN through port 514 (rsh)
Opening DKA100:
_RCD1: created
$ mount my_cd /override=id
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, OPENVMS062   mounted on _ALTARF$RCD1:
$ dir my_cd:[0,0]
```

7. RCP: Copying Files

Introduction

The Remote Copy Program (RCP) is a command you can use to copy files between your local OpenVMS host and a remote host.

Use the RCP command to copy remote files. You can copy files:

- From a remote host to your host
- From your host to a remote host
- From one remote host to another remote host (a "third-party" copy)

Before you use RCP, your system manager must install and enable the shell service during TCPware R Services configuration.

Also, make sure your host or username is registered in the remote system's `~/.rhosts` (if UNIX) or `SYS$LOGIN:.RHOSTS` file (if OpenVMS).

If you need to preserve case for any of the command elements, enclose each in quotes, since RSH lowercases unquoted text strings. Include a pair of quotes for each redirection of the command. If you are redirecting a command through one remote host to have it executed on a third, each host in turn strips off a pair of quotes after interpreting the command. In this case, you may need three pairs of quotes around the command element to preserve case.

RCP

Copies files between the local and remote host, or between two remote systems.

Format

`RCP source destination`

Parameters

source

Source host and pathname information, in the general format *host*`:`*filespec*

- *host* is the remote host name followed by a colon (`:`).
- *filespec* is different for UNIX and OpenVMS systems:
  - For UNIX system source hosts, use the absolute pathname (such as `/home/user/file.txt`) or the one relative to the user's home directory (`file.txt`).
  - For OpenVMS source hosts, use the format `[`*dir*`]`*file*`.`*typ* or *file*`.`*typ*, which assumes the current directory.

If you include a *username* or *device*, use the following format:

```
"username@host:device:filespec"
```

If you include a username and want to copy from a remote host, the remote host must include your host (and username) in its host equivalence file. If you do not use the above format, use the `/USER`, `/PASSWORD`, and `/TRUNCATE` qualifiers.

> **Note:** Do not use `/USER` or `/PASSWORD` when using DECnet syntax for a source or destination. Instead, use:
>
> ```
> host"username password"::filespec
> ```

You also cannot use DECnet syntax for both source and destination (as for a remote-to-remote copy) that involves two passwords.

destination
Destination host and pathname information, in the same format as source.

Qualifiers

/AUTHENTICATION*[=auth-type]*
Determines the authentication method. If `NULL` (or you omit the qualifier), uses standard authentication.

/LOG
Logs the files copied to or from the local system. The default is not to log. Logging only applies to the first remote host transaction in a third-party copy.

/VMS[={MULTINET | TCPWARE}]
/NOVMS
If `/VMS` is omitted, RCP by default attempts a TCPware style VMS mode transfer. This retains VMS file attributes across copies. Use `/VMS=MULTINET` to do a transfer involving a MultiNet machine. Use `/NOVMS` only if you get the error `%DCL-W-IVKEYW, unrecognized keyword - check validity and spelling` with the RCP command. `/NOVMS` disables maintaining VMS file attributes during a third-party copy.

/PASSWORD=*remote-password*
Password for the remote account. Use with the `/USER` qualifier. Do not use with DECnet source or destination syntax.

CAUTION! The password is sent across the network as plain text.

/PRESERVE
Preserves the file protection mode and modification date during a copy.

/RECURSIVE
Recursively copies each subtree rooted at the directory you specify in the UNIX system *filespec*. This makes it possible to copy entire UNIX system directories and their files. In OpenVMS, specify [*dir*...] (with the three trailing dots) in the *filespec* instead of using /RECURSIVE.

/USER=*remote-username*
User on the remote host. Use only if the remote host's ~/.hosts or .HOSTS file does not include your local host name or username. If necessary, truncate username to the required number of characters using the /TRUNCATE qualifier. Converted to lowercase if not enclosed in quotes. Do not use with DECnet file syntax.

/TRUNCATE*[=n]*
Truncates the username to the specified n number of characters, since some UNIX systems restrict the length of usernames. If you omit *n*, the default is eight characters.

Examples

1. This command copies a remote UNIX system source file in its home directory to a local host file of the same name in the current directory. The copy preserves the source file's protection mode and modification date.

```
$ rcp /preserve unixhost:src_file []
```

2. This command copies the complete remote UNIX system directory tree ~/src_dir to the local subdirectory DST_DIR, while logging the copy of each file:

```
$ rcp /recursive/log unixhost:src_dir [.dst_dir...]
```

3. The first of these two commands only copies the .src_dir subdirectory to a UNIX system. The second command copies the whole subtree.

```
$ rcp /recursive [.src_dir] unixhost:dst_dir
$ rcp [.src_dir...] unixhost:dst_dir
```

4. This command copies the complete local subdirectory tree *SRC_DIR* to a remote OpenVMS host's destination directory while preserving the directory hierarchy:

```
$ rcp [src_dir...] vmshost:[dst_dir...]
```

5. This command copies all files under the local *SRC_DIR* directory to a remote OpenVMS host's destination directory. This does not preserve the copied directory's hierarchy:

```
$ rcp [src_dir...] vmshost:[dst_dir]
```

6. This command copies all directories and files under the local *SRC_DIR* directory to a remote OpenVMS host user's login directory on the DKA300: device (use the double quotes):

```
$ rcp [src_dir...] "vmshost:dka300:[login...]"
```

7. This command copies the local *SRC_FILE* on device DKA100: to *dst_file* on a remote host. Double quotes are needed to specify a device name. The /NOVMS qualifier allows RCP to copy compatibly to an OpenVMS host running TCP/IP Services for OpenVMS (UCX).

```
$ rcp /novms ":dka100:[src_dir]src_file" ucx_host:dst_file
```

8. This command copies the local *SRC_FILE* to *~some/dst_file* if the remote host is a UNIX system, or [*some-login-directory*]*DST_FILE* if the remote host is OpenVMS. (RCP truncates the someone username to some.) In this case, the remote host does not have a host equivalence file entry for the local host, requiring /USER and /PASSWORD.

```
$ rcp /user=someone/pass=password/truncate=4 src_file host:dst_file
```

9. This command copies a file from one remote host to another (a "third-party" copy):

```
$ rcp remotehost1:file1 remotehost2:file2
```

10. Using the /USER or /PASSWORD qualifier with DECnet syntax is not allowed and returns the error message shown:

```
$ rcp /user=user1 new.txt flower"user2 password"::new.txt
TCPWARE-E-NOQUAL, /USERNAME qualifier not allowed with DECnet syntax
```

11. Using multiple passwords with DECnet syntax is not allowed and returns the error message shown:

```
$ rcp tree"user1 pass1"::new.txt flower"user2 pass2"::new.txt
TCPWARE-E-MULTPW, Multiple passwords not supported
```

8. RLOGIN: Logging in to a Remote Host

Introduction

RLOGIN is the Berkeley R Command utility you can use to log in to a remote host. RLOGIN provides a functionality similar to TELNET except that RLOGIN follows more of a UNIX format.

This chapter is a basic use summary of the RLOGIN command.

Before you use RLOGIN, be sure your host or username is registered in the remote system's `~/.rhosts` file (if UNIX) or `SYS$LOGIN:.RHOSTS` file.

See the *TCPware Management Guide*, Chapter 16, *Managing R Commands*, for information on host equivalence files.

To close an RLOGIN connection, simply log out of the remote system.

RLOGIN

Logs in to a remote host from your local host without entering a remote username and password. The remote host must provide the `login` service.

You can log in to the remote host with a different username by specifying the `/USER` qualifier.

When RLOGIN starts up, it processes the flow control characters `Ctrl+S` and `Ctrl+Q` locally unless the remote host instructs otherwise. RLOGIN passes all other keystrokes directly to the remote process and perform according to conventions established on the remote host.

The special RLOGIN commands in the below table are available once you start the connection to the remote host. Enter the special RLOGIN commands as the first character on a line.

Command	Purpose
`~.`	Closes the connection and exits RLOGIN.
`~^Z`	Spawns a subprocess on the local host and connects `SYS$INPUT`, `SYS$OUTPUT`, and `SYS$ERROR` to that process. When the subprocess logs out, control returns to the remote session. Note that you cannot spawn with CAPTIVE accounts.
`~~`	Sends a single tilde to the remote system.

Format

`RLOGIN` *`host`*

Parameter

`host`
Name or internet address of the remote host where you want to log in.

Qualifiers

/EIGHTBIT
Accepts eight-bit data from the terminal and sends it to the remote system. The default is that only seven-bit data is sent.

/ESCAPE_CHARACTER=*char*
New escape character for issuing special RLOGIN commands. The default escape character is the ~ (tilde) character.

To close your session from your local host, use a period (.) as the escape command.

/LOG=*file*
Logs a copy of the output to the specified file. Output continues to be directed to SYS$OUTPUT while it is being recorded in the log file. The default is no logging.

/LOWERCASE
/NOLOWERCASE
/LOWERCASE sends your local username to the remote host in lowercase (the default).
/NOLOWERCASE preserves any uppercase characters in the local username.

/TERMINAL_SPEED=*baud*
Terminal speed in baud rate. The default is the current speed of your terminal.

/TERMINAL_TYPE=*type*
Resets the current terminal type to the specified *type*. The allowable types you can use to override the current type are VT100, VT200, VT300, and VT400.

The remote terminal type is the same as the local terminal type. If the terminal's virtual size (rows, columns, or pixels) changes during the RLOGIN session, RLOGIN provides the remote host with the new information.

/TRUNCATE*[=n]*
Truncates the local OpenVMS username to *n* number of characters. The *n* value must be greater than zero or the command aborts with an error. The default is eight characters.

If the local username is also the remote username (if you omit the /USER qualifier), TCPware also truncates the remote username to the indicated length. However, it never truncates a remote username specified explicitly with the /USER qualifier.

/USER=*remote-username*
Username on the remote host that is different from the username with which you are currently logged in to the local host. TCPware never truncates an explicitly specified remote username (see the /TRUNCATE qualifier). The *remote-username* is converted to lowercase unless you enclose it in quotes or use the /NOLOWERCASE qualifier.

Example

Each of these equivalent commands opens a connection to host IRIS using standard authentication:

```
$ RLOGIN IRIS
$ RLOGIN /AUTH=NULL IRIS
```

9. RSH: Issuing Commands on a Remote Host

Introduction

RSH is the Berkeley R Command utility you can use to execute a single command on a remote host without logging in. This chapter is a summary of using the RSH command.

Before you use RSH, make sure your host and/or username is registered in the remote system's ~/.rhosts file (if UNIX) or SYS$LOGIN:.RHOSTS file (if OpenVMS) . See the *TCPware Management Guide*, Chapter 16, *Managing R Commands*, for details on host equivalence files.

RSH

Executes a single command on a remote host. The remote host must provide command execution service.

When the command completes execution on the remote host, the RSH command exits and closes the connection; you return to your local working environment.

RSH writes any output from the command to SYS$OUTPUT; it writes any error from the command to SYS$ERROR, unless overridden with the /OUTPUT or /ERROR qualifier.

Some servers (such as UNIX servers) send output with only line feeds for screen display. To satisfy OpenVMS screen displays, RSH inserts a carriage return by default before each line feed before sending the output to the terminal. If your screen display requires only a line feed, use the /RAW qualifier to bypass the default.

If you need to preserve case for any of the command elements, enclose each in quotes, since RSH lowercases unquoted text strings. Include a pair of quotes for each redirection of the command. If you are redirecting a command through one remote host to have it executed on a third, each host in turn strips off a pair of quotes after interpreting the command. In this case, you may need three pairs of quotes around the command element to preserve case.

Format

RSH *host command*

Parameters

host
Name or internet address of the host you want to execute the command on. Can be a domain-style name or an IP address.

command
Name of the command or command string to execute on the remote host.

Qualifiers

/ERROR=*file*

File or device to which to direct error messages from the remote command. The default is /ERROR=SYS$ERROR. (See also the /SYSERROR qualifier.)

/LOG=*file*
Logs a copy of the output to the specified file. Output continues to be directed to SYS$OUTPUT while it is being recorded in the log file. Not valid with /SYSERROR. The default is no logging.

/OUTPUT=file
Output file or device to which to direct output from the command. The default is /OUTPUT=SYS$OUTPUT.

/PASSWORD=*remote-password*
Password for the remote account. Use together with the /USER qualifier. The password is sent across the network as plain text.

/RAW
/NORAW
Prevents an extra carriage return from being inserted for screen display. Specifying /NORAW, the default, or omitting the qualifier places a carriage return before a line feed character before the line is written to the terminal.

/SYSERROR
Same as the /ERROR qualifier except that it sends messages to the NLA0 device.

/TRUNCATE*[=n]*
Truncates the local OpenVMS username to the specified *n* length. The *n* value must be greater than zero or the command aborts with an error. The default is eight characters.

If the local username is also the remote username (if you omit the /USER qualifier), TCPware also truncates the remote username to the indicated length. However, it never truncates a remote username specified explicitly with the /USER qualifier.

/USER=*remote-username*
Remote host's username that is different from the username with which you are currently logged in to the local host. TCPware never truncates an explicitly specified remote username (see the /TRUNCATE qualifier). *remote-username* is converted to lowercase unless you enclose it in quotes.

Examples

1. This command opens a connection to host IRIS and displays the name of your current working directory:

```
$ rsh iris pwd
```

2. This command opens a connection to host IRIS for username "Smith" and displays the name of the working directory for "Smith":

```
$ rsh iris /user="Smith" pwd
```

The quotes around `Smith` are necessary because the name contains a mixture of upper- and lowercase characters that you would want to preserve in sending the command. Without the quotes, the name converts to lowercase and may not match the username on the remote host.

3, This command opens a connection to host IRIS and displays the name of your working directory in a "raw" state on a terminal that requires only line feeds to display the information:

```
$ rsh iris /raw pwd
```

4. This command executes a `pwd` command on ROSES as sent through VIOLET.

```
$ rsh violet /user=system /password=plastic -
_$ rsh roses /user=root/password="""TCPware""" pwd
```

The TCPware password is triple-quoted to preserve case through the transaction. The system strips off the first pair of quotes and executes `rsh roses /user=root/pass=""TCPware""`. VIOLET strips off the second set of quotes and executes `rsh roses /user=root/pass="TCPware".` ROSES strips off the third and executes `pwd`. In each case, the password string is interpreted literally.

10. Sending and Receiving Electronic Mail

This chapter describes how to use OpenVMS MAIL and ALL-IN-1 Mail with TCPware and covers the following major topics:

- Using OpenVMS mail across the network
- Using mail under ALL-IN-1 across the network

Using OpenVMS Mail across the Network

TCPware enhances OpenVMS Mail so you can send and receive mail across the network.

Specifying Addresses

When you use OpenVMS Mail to send mail to a host outside your VMScluster, the message is sent via SMTP (Simple Mail Transfer Protocol). For this reason, you must specify the address so that SMTP accepts the mail correctly. The format for the address is:

`To: SMTP%"`*`recipient@destination`*`"`

The string `SMTP` and the destination system name are not case-sensitive; that is, you can type them in either uppercase or lowercase letters. The destination recipient specification may be case- sensitive, however, depending on the destination system's software. On some UNIX systems, ROOT and root specify two different usernames (and hence different electronic mail addresses).

If the address contains a quote, enter the address with either `\'` or `\s` as shown in the following example formats:

`To: SMTP%"\'`*`recipient@destination`*`"`

or

`To: SMTP%"\s`*`recipient@destination`*`"`

If the address is on a local DECnet network, use this format:

`To: SMTP%`*`nodename`*`::`*`username`*

If the address is on a remote DECnet network, you may use this format:

```
To: SMTP%"'nodename::username'@destination"
```

> **Note:** TCPware assumes that an address containing a double colon (`::`) is a DECnet address. If an address contains a double colon and is not a DECnet address, SMTP does not handle it correctly.

If you know the recipient's IP address, but not the host name (or if the host name is not registered in the Domain Name System), specify the recipient address as follows:

```
To: smtp%"recipient@[aa.bb.cc.dd]"
```

Where *aa.bb.cc.dd* is the destination system's IP address in dotted-decimal form. You must specify the IP address in square brackets.

The OpenVMS Mail utility also allows you to specify an addressee on the command line:

```
$ MAIL filename addressee
```

To use this form of the command with TCPware, you must enclose the address in quotes (and you must double all existing quotes), as follows:

```
$ MAIL filename "smtp%""recipient@destination"""
```

The following example shows the user sending mail using the OpenVMS MAIL utility to a user named John Smith with a username of "johns" on system SALES.EXAMPLE.COM.

```
$ MAIL
MAIL>SEND
To:     SMTP%"johns@sales.example.com"
Subj:   This is a test message.
Enter your message below. Press Ctrl/Z when complete, or
Ctrl/C to quit:
Hi John, this is a test of the TCPware extension to the VMS MAIL
utility.
Ctrl+Z
MAIL>EXIT
$
```

You receive network mail as you would all other mail in the VMS MAIL utility. The following example shows the user "WHORFIN" reading an SMTP mail message sent by the user "johns."

```
$
New mail on node KAOS from SMTP%"johns@sales.example.com" "John Smith"
$ MAIL
You have 1 new message.
MAIL>READ/NEW
#1          03-23-2021 10:05:40.79
From:   SMTP%"johns@sales.example.com"      "John Smith"
To:     WHORFIN
CC:
Subj:   Re: This is a test message.

Return-Path: <system@karem.example.com>
Received: from karem.example.com (192.168.1.92) by dino.example.com
          (MX V5.1-X A2w8g) with SMTP for <smith@paul.example.com>;
          Mon, 9 Aug 2021 14:35:01 -0400
Received: by karem.example.com for smith@water.example.com;
          Mon, 9 Aug 2021 14:35:00 GMT
Date: Mon, 9 Aug 2021 14:35:00 GMT
From: system@karem.example.com
To: smith@water.example.com
Message-ID: <990809143500.a2@karem.example.com>
Glad to see your test worked.
This is my response.

MAIL>EXIT
```

Specifying a Host Alias

TCPware allows a system to have multiple names - or *host aliases* - with respect to electronic mail delivery. You can specify the host alias you want to use by defining the `TCPWARE_SMTP_FROM_HOST` logical name. The alias you choose must be one of the SMTP host name aliases registered on the system (see the translation of the logical name `TCPWARE_SMTP_HOST_NAME` and the contents of the file `TCPWARE_HOST_ALIAS_FILE`). If the alias you use is unknown, the setting of `TCPWARE_SMTP_FROM_HOST` is ignored.

The host alias feature allows users from different administrative units within an organization to have their return address reflect the name of their unit, even though mail for all units is handled by one system.

Specifying Individual Aliases

TCPware supports both *system-wide* and *per-user* mail aliases. Using these aliases, you can refer to electronic mail addresses with names that are meaningful to you. Per-user mail aliases are kept in the file `SMTP_ALIASES.` in your login directory.

The format for alias entries is:

```
alias:      real_address[,...];
```

where *alias* is an alphanumeric string and *real_address* is an 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 may have a username of JB134A, but you want to send mail to him as john. Add the following line to your SMTP_ALIASES. file:

```
john:       jb134A;
```

Aliases are repeatedly translated until no more translations are found. You can circumvent the repeated translations by including a leading underscore (_) in the *real_address*. For example, this definition causes mail to be forwarded and delivered locally:

```
fnord:      fnord@somewhere.example.edu, _fnord;
```

Using Mail under ALL-IN-1

This section explains how to use the mail subsystem under ALL-IN-1 to send mail to and receive mail from users on remote systems.

To send mail to a user on a remote system, specify an ALL-IN-1 e-mail address in the format:

```
recipient@destination@SMTP
```

@SMTP indicates to the ALL-IN-1 mail subsystem that the message should be given to the SMTP/MR gateway facility for eventual handling by the TCPware SMTP mail system. Note that the string SMTP and the destination system name are not case-sensitive; that is, you can type them in either uppercase or lowercase letters. However, the destination recipient specification may be case-sensitive, depending on the destination system's software. On some UNIX systems, ROOT and root specify two different user names (and hence different electronic mail addresses).

You receive network mail as you would all other mail in the ALL-IN-1 mail subsystem. Contact your system manager for the correct syntax for remote users; frequently, the proper syntax is:

```
yourname@A1.yourdomain
```

Delivering Mail to Specific Folders

The SMTP server supports incoming mail delivery to folders other than the NEWMAIL folder. The folder names are restricted to UPPERCASE characters only, the pound sign (#), and the underscore (_). Use of the comma (,) in a folder name causes an error. Mail addressed to *user+folder@host* is delivered to the specified *folder*.

> **Note:** Your system manager can disable this feature by defining the system-wide logical name TCPWARE_SMTP_DISABLE_FOLDER_DELIVERY.

User-Defined Headers

You can further customize your messages by defining special RFC 822 message headers.

TCPware SMTP supports defining certain message header fields in the RFC 822 part of the message header.

Defining RFC 822 headers involves running the TCPWARE:CONFIG_SMTP_HEADERS.COM command file to define the following headers:

- Full name
- Comments
- Reply-to
- Return-receipt-to
- Bcc
- Sender
- X-Department
- X-Special user-defined header

Run the command procedure:

```
$ @TCPWARE:CONFIG_SMTP_HEADERS
```

The procedure checks for the TCPWARE_SMTP_USER_HEADERS logical for header definitions. If it does not find the logical, it checks the SYS$LOGIN:SMTP_USER_HEADERS.COM file. If it finds the file, it comes back with the prompt:

```
SYS$LOGIN:SMTP_USER_HEADERS.COM Exists.  Load? [Yes]
```

If you want to accept the contents of the file, press **Return**. (If the file did not load properly, you can have it overwritten at the next prompt.) You then have the choice of adding to, modifying, or deleting the file, exiting and saving, or quitting without saving:

```
[A]dd, [M]odify, [D]elete, e[X]it and Save or [Q]uit:
```

If you are adding a header, the following prompt appears:

```
Add Header:
1. Full-Name:
2. Comments:
3. Reply-To:
4. Return-Receipt-To:
5. Bcc:
6. Sender:
7. X-Department:
8. Other
Which header would you like to add?
```

Enter the number value:

`1` - Enter your full name
`2` - Enter a comments line
`3` - Enter a reply-to name address
`4` - Enter a return-receipt-to name or address

A return-receipt-to value is only valid if the system logical `TCPWARE_SMTP_RETURN_RECEIPT_TO_HEADER_ENABLE` is defined as `1` during configuration. If this system logical is not defined or defined as `0`, TCPware does not add the `Return_receipt_to` header to the mail message.

`5` - Enter a Bcc: name or address
`6` - Enter a sender name or address
`7` - Enter a departmental name or address

TCPware prepends an `X-` to the departmental name or address.

`8` - Enter your own special header. For example:

```
What is the name of the header: X-Affiliation
```

TCPware prepends an `X-` to the special header name.

The next prompt asks you to supply a value for the header you specify. For example:

```
Full-Name Value: George Plimpton
```

The procedure returns to the `[A]dd, [M]odify, [D]elete, e[X]it and Save or [Q]uit:` prompt so that you can add other headers or modify or delete existing ones. If you enter `X` (exit and save), the procedure writes out the file on exiting and defines the `SMTP_USER_HEADERS` logical based on the file's contents.

If you are modifying a header definition, the procedure gives you the current list of defined headers, followed by a prompt, where you enter the appropriate number. For example:

```
Your Current Headers:
1. Full-Name Value: George Plimpton
2. X-Affiliation: Paris Review
Which header would you like to modify? 2
New X-Affiliation Value: None
```

After modification, you return to the `Which header would you like to modify?` prompt. If you enter **Return** at the prompt, you return to the `[A]dd,...` prompt.

If you are deleting a header definition, the procedure gives you the current list of defined headers, followed by the prompt:

```
Which header would you like to remove?
```

The procedure asks for confirmation and returns to the above prompt unless you enter **Return**. Removed files show up as being deleted in the `Your Current Headers:` list until you add a new header or exit and reenter the procedure.

11. TALK: Exchanging Terminal Messages

Introduction

The TALK utility allows you to exchange messages you type at your terminal with another local or remote user. You do not need to wait between sending your message and receiving one from your destination user. TALK uses a split screen where what you type is on the top half and what the other person types is on the bottom. This allows you to talk in "real time."

Using TALK

First make sure the OpenVMS Phone utility is on. If you show the broadcast status for your terminal and get something like the following:

```
$ SHOW BROADCAST
Broadcasts are currently disabled for:
    PHONE
    MAIL
    QUEUE
    SHUTDOWN
```

Then you enable phone broadcasting as follows:

```
$ SET BROADCAST=PHONE
```

To set up and invoke TALK, enter at the DCL prompt:

```
$ TALK:==$TCPWARE:TALK.EXE
$ TALK username[@host] [ttyname]
```

If you are communicating with another local user, type the user's *username*. If communicating with a user on another system, use the *username*@*host* syntax.

You can also include the terminal port (*ttyname*) as a parameter. Most UNIX servers only ring one of and not all the remote user's terminals. If the remote user is logged in many times, and you would rather ring a terminal that has been idle for only a short period, specify the terminal port using *ttyname*.

One way to discover terminal ports is by using the FINGER utility, such as in the following example, where there are two terminal ports, `ttyp5` and `ttyp7`. Since the `ttyp7` terminal has a much shorter idle time (and is more current), it is therefore a better candidate for a TALK terminal:

```
$ FINGER MARGE@MARGE.EXAMPLE.COM
Login name: marge                       In real life: Marge Simpson
Directory: /home/spectre                Shell: /usr/local/bin/tcsh
On since Nov 3 10:06:48 on ttyp5 from bart.example.com
59 minutes Idle Time
Login name: marge                       In real life: Marge Simpson
Directory: /home/spectre                Shell: /usr/local/bin/tcsh
On since Nov 3 10:06:44 on ttyp7 from bart.example.com
36 seconds Idle Time

$ TALK MARGE@MARGE.EXAMPLE.COM TTYP7
```

After the above command, TALK sends the following message to the recipient if the connection is successful:

```
Message from Talk_Daemon@destination-host...
talk: connection requested by yourname@yourhost.
talk: respond with: talk yourname@yourhost
```

To establish the connection, the recipient follows the instructions from the Talk_Daemon and types the following at the system prompt:

```
$ talk yourname@yourhost
```

It does not matter from which machine the recipient replies, as long as the recipient's login name is the same. Once communication is established, the two parties can type simultaneously, with their output appearing in two parts of a split screen. What you type appears on the top half and what the other person types is on the bottom half of the screen.

To signal that you are expecting a response, it is customary to leave a blank line after your last line of text. You can use a convention such as **-oo** ("over and out") to signal that your part of the correspondence is over.

Type `Ctrl+L` to reprint the screen. You can also use the erase, kill, and word kill (`Ctrl+K`) characters.

To exit, type the interrupt character (`Ctrl+C`, `Ctrl+Y`, or `Ctrl+Z`). TALK moves the cursor to the bottom of the screen and restores the terminal to its previous state.

The below example shows a sample exchange between user BART on host BART.EXAMPLE.COM (an OpenVMS system) and user MARGE on host MARGE.EXAMPLE.COM.

On Bart:

```
(Bart) $ TALK:==$TCPWARE:TALK.EXE
(Bart) $ TALK MARGE@MARGE.EXAMPLE.COM
------------[Waiting for your party to respond]---------------------
```

On Marge:

```
(Marge) $
Message from Talk_Daemon@BART.EXAMPLE.COM at 11:23 ...
talk: connection requested by bart@bart.example.com.
talk: respond with:  talk bart@bart.EXAMPLE.com
(Marge) $ TALK BART@BART.EXAMPLE.COM
------------[Connection established: bart@bart.example.com]-----------
```

On Bart:

```
Hi, there!
------------[Connection established:
marge@marge.example.com]-----------
```

On Marge:

```
Good to hear from you!
------------[Connection established: bart@bart.example.com]--
Hi, there!
```

On Bart:

```
Hi, there!
See you soon! -oo   Ctrl+C
------------[Connection closed.
Exiting]----------------------------
Good to hear from you!
(Bart) $
```

Command Reference

The following is a command reference to the TALK utility.

TALK

The TALK command is a visual communication program that exchanges messages with another host user by copying lines you type on your terminal to the other user's terminal. The other host recipient must support the ntalk protocol to accept (and respond to) your messages.

It does not matter from which machine the recipient replies, as long as the recipient's login name is the same. Once communication is established, the two parties can type simultaneously, with their output appearing in different parts of the same window.

Typing **Ctrl+L** causes the screen to be reprinted, while the erase, kill, and word kill (**Ctrl+K**) characters work in TALK as normal.

To exit, type your interrupt character (**Ctrl+C**, **Ctrl+Y**, or **Ctrl+Z**). TALK moves the cursor to the bottom of the screen and restores the terminal.

Format

TALK *username*[@*host*] [*ttyname*]

Parameters

***username*[@*host*]**

If you want to talk to someone on your own machine, *username* is just the local user's login name. If you want to talk to a user on another host, use the form *username*@*host*.

ttyname

Name of the specific remote terminal. Many UNIX clients do not send talk request messages to every terminal of the user, and usually select just one. You may, however, want to make a particular selection.

Restrictions

TALK is not eight-bit clean. Typing in DEC Multinational Characters (ISO-8859/1) causes the characters to echo as a sequence of carets (^) followed by the character represented with its high bit cleared. This limitation makes TALK unusable if you want to communicate using a language that has ISO-8859/1 characters in its alphabet.

Example

```
system1>talk user2@system2
```

The following message appears on the screen of `user2`:

```
Message from Talk_Daemon@system2 at 12:37 ...
talk: connection requested by user1@system1.
talk: respond with: talk user1@system1
```

To establish the connection, `user2` follows the instructions from the `Talk_Daemon` and types the following at the system prompt:

```
system2>talk user1@system1
```

Troubleshooting

The `Your party is refusing messages` message may come up if the remote terminal is set up with messages off, such as the first terminal (`tty5`) in the following example:

```
(Bart) FINGER MARGE@MARGE.EXAMPLE.COM
Login name: marge      (messages off)    In real life: Marge Simpson
Directory: /home/spectre                 Shell: /usr/local/bin/tcsh
On since Nov 3 10:06:48 on ttyp5 from bart.example.com
59 minutes Idle Time
No unread mail
No Plan.

Login name: marge                        In real life: Marge Simpson
Directory: /home/spectre                 Shell: /usr/local/bin/tcsh
On since Nov 3 10:06:44 on ttyp7 from bart.example.com
36 seconds Idle Time
(Bart) TALK MARGE@MARGE.EXAMPLE.COM TTYP5
----------------[Your party is refusing messages]--------------------
```

The `Checking for invitation on caller's machine` message may come up when the client is waiting for a response from the remote system. If the message appears for an extended time, it may mean that the remote system's server does not support the ntalk protocol, in which case a connection is not possible.

If the message `Your party is not logged on` appears, the remote user is not logged on at the time.

12. TELNET: Connecting to Remote Terminals

Introduction

The Virtual Terminal Protocol (TELNET) provides connections to remote hosts. With it, you can access remote hosts using OpenVMS commands or a UNIX style command interface.

You can run the Telnet client interactively or through a startup command procedure.

Before Using TELNET

Before you use TELNET, ask your system manager if the TELNET software was installed, configured, and started on your system.

- Before you can connect to a remote host, you need to know:
- The name of the remote host to which you want to connect.
- The username and password for each account on the remote host. If the remote host does not support multiuser protection features, you may not need a username and password.
- How to use the operating system of the remote host.

> **Note:** The Telnet client does not restrict the ASCII character set to seven-bit ASCII as the TELNET standard implies. The Telnet client supports the full eight-bit (multinational) character set. To use the multinational character set, you must configure your terminal to support eight-bit characters. The peer TELNET implementation must also support the same.

Opening a TELNET Session

Run the Telnet client utility to connect to a remote host. The Telnet client supports as many as 10 connected sessions at any one time. To open a TELNET session (see the below example):

1. At the DCL prompt, enter:

```
$ TELNET
```

   a. Use the OPEN command to open a remote TELNET session. At the TELNET> prompt, enter:

```
TELNET>OPEN host
```

   host is the name of the host to which you want to connect.

2. Respond to the login prompts, if any, of the remote host.
3. Open another session if desired:
   a. Return to the local TELNET> prompt by entering the escape sequence displayed when opening the connection (usually **Ctrl+**). The previous session remains open.
   b. Use the OPEN command to open the next session. Repeat steps 2 and 3.

You can also open a remote TELNET connection as follows:

```
$ TELNET host
```

Example showing the opening of multiple telnet connections:

```
(IRIS) $ TELNET
TELNET>OPEN BART

%TCPWARE_TELNET-I-TRYING, trying bart.example.com,telnet(192.168.1.92,23)...
%TCPWARE_TELNET-I-ESCAPE, escape (attention) character is "^\"

(BART) $ Ctrl+\

TELNET> OPEN MARGE                    [BART remains open]

%TCPWARE_TELNET-I-TRYING, trying marge.example.com,telnet
(192.168.1.91,23)...
%TCPWARE_TELNET-I-ESCAPE, escape character is "^\"

(MARGE) $ Ctrl+\

TELNET>OPEN HOMER                              [BART and MARGE remain open]

%TCPWARE_TELNET-I-TRYING, trying homer.example.com,telnet
(192.168.1.90,23)...
%TCPWARE_TELNET-I-ESCAPE, escape character is "^\"

(HOMER) $ Ctrl+\

TELNET> OPEN LISA     [BART, MARGE, and HOMER remain open]
```

```
%TCPWARE_TELNET-I-TRYING, trying lisa.example.com,telnet
(192.168.1.89,23)...
%TCPWARE_TELNET-I-ESCAPE, escape character is "^\"

(LISA) $ Ctrl+\

TELNET>
```

Closing a Session

A TELNET session remains open until you log out of that session at the system prompt or use the CLOSE, EXIT, QUIT, or BYE commands or enter **Ctrl+Z** at the TELNET> prompt.

To close a TELNET session, use one of the following commands at the TELNET> prompt (see the below example):

- TELNET>**CLOSE** closes the current session, as in the following chart:

If you open a TELNET session using...	And...	Then CLOSE closes the current session and...
TELNET>OPEN *host*	It is the only session	Keeps you in TELNET
	There are other sessions	Keeps you in TELNET with the other sessions open
$ TELNET *host*	It is the only session	Exits TELNET
	There are other sessions	Keeps you in TELNET with the other sessions open

  If you close the current session, and there are other connected sessions, the Telnet client resets the current session to the "next" session.

- TELNET>**CLOSE *session-number*** closes only the specified session, as indicated by the SHOW STATUS command.
- TELNET>**EXIT** exits TELNET
- TELNET>**QUIT** exits TELNET
- TELNET>**BYE** exits TELNET
- TELNET>**Ctrl+Z** interrupts TELNET

```
(IRIS) $ TELNET
TELNET>OPEN BART

%TCPWARE_TELNET-I-TRYING, trying bart.example.com,telnet(192.168.1.92,23)...
%TCPWARE_TELNET-I-ESCAPE, escape character is "^\"

(BART) $ Ctrl+\
TELNET> OPEN MARGE      [BART remains open]

%TCPWARE_TELNET-I-TRYING,trying marge.example.com,telnet(192.168.1.91,23)...
%TCPWARE_TELNET-I-ESCAPE, escape character is "^\"

(MARGE) $ Ctrl+\

TELNET>SHOW STATUS

The Telnet client V6.1-0 Copyright (c) Process Software
Connected sessions:
     1. bart.example.com,telnet (192.168.1.92,23).
 --> 2. marge.example.com, telnet (192.168.1.91,23).
"^\" is the escape (attention) character

TELNET> CLOSE 2

%TCPWARE_TELNET-I-CONNCLOSED, closing session 2, marge.example.com

TELNET>CLOSE 1

%TCPWARE_TELNET-S-CONNCLOSED, closing session 1, bart.example.com

TELNET>EXIT
(IRIS) $
```

Issuing Local Commands

You can issue commands to the Telnet client utility during a remote session by returning to the TELNET> prompt. You can then enter one or more TELNET commands.

TELNET OpenVMS features multiline recall of up to 20 command lines using the standard OpenVMS line recall and editing keys.

You return to the remote session by entering the RESUME command.

To issue a local TELNET command while connected to a remote host and then resume the session on the host (see the below example):

1. Enter the escape (attention) character to return to the TELNET prompt: for example: **Ctrl+**

2. Issue a TELNET command. For example, you may want to:
   - Issue the `SHOW STATUS` command. The `SHOW STATUS` command displays a list of open connections. The arrow (`-->`) identifies the current session.
   - Change the escape (attention) character using the `SET ESCAPE` command.
3. Return to the remote host by entering:

   ```
   TELNET>RESUME
   ```

   This command resumes to the current remote host. Pressing **Return** or entering the `OPEN` command also resumes to the current remote host.

   To resume to a different session, enter:

   ```
   TELNET>RESUME session-number
   ```

   `session-number` is the number of the session which you want to resume. The session-number refers to a particular connection, as displayed by the `SHOW STATUS` command.

You can switch between local TELNET command mode and the remote host as often as you like.

```
(BART) $ Ctrl+\

TELNET>SHOW STATUS

Telnet client V6.1 Copyright (c) Process Software
Connected sessions:
           1. BART.example.com,  telnet  (192.168.1.92,23).
           2. HOMER.example.com, telnet  (192.168.1.90,23).
           3. MARGE.example.com, telnet  (192.168.1.91,23).
       --> 4. LISA.example.com,  telnet  (192.168.1.89,23).

"^\" is the escape (attention) character.

TELNET>SET ESCAPE "^A"

escape (attention) character is "^A"

TELNET>RESUME

(BART) $
(BART) $ Ctrl+\

TELNET>RESUME 2

%TCPWARE_TELNET-I-RESUME, resuming session 2, HOMER.example.com

(HOMER) $
```

Running Applications over TELNET

You can run applications over a TELNET connection by creating an NTA terminal on the local client. You can only create such devices from TELNET with no other escaped connection. This section describes how to create non-permanent NTA devices. To create permanent NTA devices, see the next section.

Normally, the Telnet client connects to an NTA device at the TCPware server end of the connection. It does not usually create a local NTA device. However, you can create a local NTA device so that you can run applications over the TELNET connection. To create a local NTA device (see the below example):

1. Enter at the DCL prompt one of the following:

   - `$ TELNET host /CREATE`
   - `TELNET>OPEN /CREATE`

   Use the second method if you already logged in to a host and escaped from the session (using **Ctrl+** or some other defined escape sequence).

   In both cases, this associates a pre-allocated local `NTA`*x*`:` terminal device to your TELNET connection; *x* is the next available unit number. No other escaped connection can exist during your TELNET session for this to work. (If one exists, the `%TCPWARE_TELNET-E-CONNOPN` error message appears.)

2. Run your application at the DCL prompt. Use the allocated terminal device as desired.
3. When your application ends, clean up by deallocating the NTA device you created using the following command at the DCL prompt: `$ DEALLOCATE device`

   See your OpenVMS documentation for details on the `DEALLOCATE` command.

> **Note:** Using `/CREATE` in this way creates a non-permanent NTA device, which has certain ramifications. See the next section for details on how to create a permanent NTA device. Using the `OPEN /CREATE` command as part of a TELNET command file creates an NTA device and exits TELNET right away without passing any further commands in the file to TELNET.

You can also invoke TELNET and use `OPEN/CREATE` non-interactively, such as with a batch file. The batch file cannot open an interactive connection. For applications run by the creating process, use the

/LOGICAL qualifier to create a predefined name for the device. If this device is to be used by another process, the qualifier /LOGICAL=... /TABLE= may help reference it. For example:

```
$ TELNET SIGMA /CREATE /LOGICAL=TELNET_NTA /TABLE=SYSTEM /MODE=EXEC
```

See the OPEN command in the *Command Reference* section of this chapter for other parameters you can use with the /CREATE qualifier.

Opening a TELNET Connection to a Terminal Device:

```
$ TELNET MARGE /CREATE

%TCPWARE_TELNET-I-TRYING, trying marge.example.com,telnet
(192.168.1.91,23)...
%TCPWARE_TELNET-I-ALOC,_MARGE$NTA1: allocated

$ SET HOST/DTE NTA1:
$ DEALLOCATE NTA1:

$ TELNET BART

%REM-I-TOQUIT, connection established
Press Ctrl/\ to quit, Ctrl/@ for command mode
OpenVMS 8.4 with TCPware for OpenVMS 6.1

Username: Ctrl+\

TELNET>OPEN /CREATE
%TCPWARE_TELNET-I-ALLOC, _NTA1: allocated
$ SET HOST/DTE NTA1:
$ DEALLOCATE NTA1:
```

Creating a Permanent NTA Device

You can also run applications over a TELNET connection by creating a permanent NTA terminal on the local client. This permanent device acts more like a LAT device; it is not automatically deleted when there are no process channels assigned to it, it can be handed off to other applications, and it has reconnect capabilities in case of a connection break. This section describes how to create permanent NTA devices. To create non-permanent NTA devices, see the previous section.

Using TELNET /CREATE by itself to create a non-permanent NTA device, such as in the previous section, has the following limitations:

- An application using this NTA device may be written to deassign and thus delete the device if the connection goes down. This could cause a conflict when rerunning the application if, meanwhile, another NTA connection with the same unit number is created.

- Handing off the NTA device to another process may require setting up the device as NOHANGUP.
- Recovery is not possible in case of a broken connection.

You can bypass these limitations and make the NTA device a permanent one by adding the `PERMANENT` keyword to the `TELNET /CREATE` command, as follows (see the following example):

> **Note:** Creating a permanent NTA device requires OPER privilege.

1. Enter at the DCL prompt:

   ```
   $ TELNET host port /CREATE=PERMANENT
   ```

   or:

   ```
   TELNET>OPEN host port /CREATE=PERMANENT
   ```

   This creates a permanent local `NTA`*x*`:` terminal device with the next available unit number. However, unlike non-permanent NTA devices, the TELNET utility does not pre-allocate it. Likewise, you can specify the `/LOGICAL` qualifier to set up a logical name for the device so that other applications can use it.

   It is advisable that you specify a port other than the default TELNET port 23.

   See the `OPEN` command in the *Command Reference* section of this chapter for other parameters you can use with the `/CREATE=PERMANENT` qualifier.

2. Run your application at the DCL prompt, as with a non-permanent NTA device. The difference is that handing off the NTA device to another process and recovery of a broken connection are enhanced.
3. In handing off the NTA device to another process, you may wish to change its protection:
   - In VMS 5, use `SET PROTECTION`, or `SET DEVICE /ACL`
   - VMS 6 and higher replaces these commands with `SET SECURITY /PROTECTION= /ACL`

Setting up a Permanent NTA Device:

```
TELNET>OPEN MARGE 7 /LOGICAL=MY_PORT -
_TELNET>/CREATE=(PERMANENT,INTERVAL=10,RETRIES=10)
```

```
%TCPWARE_TELNET-I-CREATED, _NTA1: created

$ @MY_APPLICATION MY_PORT
```

Handling a Broken Connection

If the connection to the remote port is broken, a temporary NTA device is reported as "Offline" with $QIO's failing with a `SS$_DEVOFFLINE` status. For a permanent NTA device, however:

- The NTA devchar is marked `UNAVAILABLE` (which can be viewed by using `SYS$GETDVI` to check if `DVI$_DEVCHAR's DEV$V_AVL` = 0).
- If a terminal `Ctrl+Y` AST is set up, the AST fires up. (Setup: Disable `Ctrl+Y` handling by DCL using `LIB$DISABLE_CTRL( &LIB$M_CLI_CTRLY, 0 )`, and set up the AST using `SYS$QIOW` with `IO$_SETMODE | IO$M_CTRLYAST`).
- Terminal I/Os queued in the TTdriver are completed with the I/O Status Block (IOSB) having a status of `SS$_HANGUP`.
- A new write $QIO buffers the data so that it can be sent when reconnected. If no reconnection is being done, then one is set up.
- Data sent at the time of the broken connection may be lost.
- The client attempts to reconnect to the remote port as described in the `OPEN /CREATE=PERMANENT` command section.
- The permanent NTA handles reconnects internally instead of allowing the program to issue the LAT `SYS$QIOW` with `IO$TTY_PORT | IO$M_LT_CONNECT`.

Closing the Connection After a De-Assign

You can use the `CLOSE_DASSGN` keyword to the `/CREATE=(PERMANENT)` qualifier to close the underlying TCP connection when the last channel assigned to the NTA device is dropped using `SYS$DASSGN`. The default is not to close the TCP connection.

Startup Command File

You can have a startup file executed each time you invoke the Telnet client. The `TELNET_STARTUP` logical specifies a file that contains commands you want performed at the beginning of each TELNET session.

To set up and run a startup command file (see the below example):

1. Create a `TELNET_STARTUP.COM` file in your login directory.

2. In the file, include the TELNET command or commands you want executed each time you start the Telnet client.
3. Edit your LOGIN.COM file and define the TELNET_STARTUP logical name to point to the startup file. For example, add the following line to your login file:

```
$ DEFINE/PROCESS TELNET_STARTUP "SYS$LOGIN:TELNET_STARTUP.COM"
```

4. Rerun LOGIN and run TELNET.

Whenever you run TELNET, it first looks for the file to which the TELNET_STARTUP logical points. It then processes all the commands contained in that file until it processes the EXIT command or reaches the end of the file.

If the OPEN command appears in this file, TELNET establishes the connection, and all further input comes from the terminal. When you return to command mode, TELNET processes the rest of the commands in the startup file (if any).

If the EXIT command appears in the startup file, the Telnet client ignores all commands following the EXIT command and continues TELNET operations, leaving the user at the TELNET prompt.

Setting Up a Startup Command File:

```
$ CREATE TELNET_STARTUP.COM
SET TRANSLATION /SEND=CR
OPEN IRIS
OPEN HOMER
SHOW STATUS
Ctrl/Z
$ EDIT SYS$LOGIN:LOGIN.COM
$ DEFINE/PROCESS TELNET_STARTUP "SYS$LOGIN:TELNET_STARTUP.COM"
Ctrl/Z
$ @SYS$LOGIN:LOGIN
$ TELNET
TELNET>SET TRANSLATION /SEND=CR
%TCPWARE_TELNET-I-TRNSNEWLN, will translate CR to CRLF when sent
TELNET>OPEN IRIS
%TCPWARE_TELNET-I-TRYING, trying IRIS.plants.com,telnet
 (192.168.1.93,23) ...
%TCPWARE_TELNET-I-ESCCHR, escape (attention) character is "^\"
(login procedure to IRIS)
(IRIS)$ Ctrl/\
TELNET>OPEN HOMER
%TCPWARE_TELNET-I-TRYING, trying HOMER.illiad.com,telnet
(192.168.1.90,23) ...
%TCPWARE_TELNET-I-ESCCHR, escape (attention) character is "^\"
(login procedure to HOMER)
(HOMER)$Ctrl/\
TELNET>SHOW STATUS
```

```
Connected sessions:
1. IRIS.plants.com, telnet          (192.168.1.93,23).
-->2. HOMER.illiad.com, telnet      (192.168.1.90,23).
"^\" is the escape (attention) character.
No characters are translated to CRLF when received.
CR is translated to CRLF when sent.
TELNET>RESUME
(HOMER)$
```

Sample Session

This section shows a sample Telnet client session.

See the below example for the corresponding numbered steps. In this sample session, a user on IRIS:

1. Starts `TELNET`.
2. Enters the `SHOW STATUS` command.
3. Connects to TULIP.
4. Logs in and does some work. Enters the escape (attention) character to return to the `TELNET` prompt.
5. Changes the escape (attention) character and enters a `SHOW STATUS` command.
6. Enters the `RESUME` command to return to TULIP.
7. Logs out of TULIP.
8. Exits `TELNET`.

Sample Telnet Session:

```
(Iris) $ TELNET
TELNET>SHOW STATUS
The Telnet client V6.1 Copyright (c) Process Software
No connection established.
Terminal type list: VT300, DEC-VT300, IBM-3278-2
"^\" is the escape (attention) character

TELNET>OPEN TULIP
%TCPWARE_TELNET-I-TRYING, trying tulip.example.com,telnet
(192.168.1.56,23)...
%TCPWARE_TELNET-I-ESCCHR, escape (attention) character is "^\"

SunOS 5.9 (tulip.example.com)(ttyp2)

login: root
Password: ********

Last login: Wed Feb 21 10:57:25 from 198.168.1.105
Sun Microsystems Inc.   SunOS 5.9       Generic May 2021
```

```
tulip>ls
bin      mnt     notes     test.c     test_def.h
tulip>^\

TELNET>SET ESCAPE "^A"
%TCPWARE_TELNET-I-ESCCHR, escape (attention) character is "^A"
TELNET>SHOW STATUS
The Telnet client V6.1 Copyright (c) Process Software
Connected session:
     --1. tulip.example.com,telnet (192.168.1.56,23).
"^A" is the escape (attention) character
TELNET>RESUME
tulip>ls -A
.              .forward    bin          test.c
..             .login      mnt          test_def.h
.cshrc         .profile    notes
TELNET>EXIT
(Iris)$
```

Command Reference

The following pages consist of command descriptions for the available Telnet client commands.

You interact with the Telnet client by typing commands at the TELNET> prompt. The Telnet client supports the following OpenVMS-style commands:

CLOSE	SET *[NO]* BINARY	SET *[NO]*LOCAL_FLOW
DEFINE/KEY	SET *[NO]*BRK	SET LOG
EXIT	SET DEBUG	SET PRINT
FLUSH	SET DELETE_ALLOWED	SET TERMINAL_TYPE
HELP	SET *[NO]*EC	SET TRANSLATION
OPEN	SET *[NO]*EL	SET *[NO]*XDISPLOC
RESUME	SET *[NO]*ESCAPE	SHOW OPTIONS
SEND	SET *[NO]*FLUSH	SHOW STATUS
SET *[NO]*AO	SET *[NO]*FORWARD	SHOW TRANSLATION

SET [NO]AYT	SET [NO]GA	SPAWN
SET [NO]BACKWARD	SET [NO]IP	

Telnet command synonyms:

Synonym	Equivalent	Synonym	Equivalent
BYE or QUIT	EXIT	SET HOST	OPEN
CONNECT	OPEN	STATUS	SHOW STATUS
DISCONNECT	CLOSE	Z	SPAWN
ESCAPE	SET ESCAPE		

CLOSE

Closes the current connection or the session specified by the session number. If you are not connected to a remote host, this command has no effect.

When you open a session using the alternate `TELNET host` format, the `CLOSE` command:

- Exits TELNET if the connection is the only session.
- Keeps you in TELNET with the other session(s) open if there is at least one other session.

Format

`CLOSE [session-number]`

Synonym

`DISCONNECT [session-number]`

Parameter

session-number

Session number to close, based on the session number displayed by the `SHOW STATUS` command. If omitted, closes the current session. If there are any other connections open, the Telnet client resets the current session to the "next" one.

Examples

You can use the `SHOW STATUS` command to display a list of open connections. These examples start with HOMER as the current session. There are three telnet connections, as follows, with the current session being on HOMER:

```
     1. BART.example.com,  telnet (192.168.1.92,23).
     2. MARGE.example.com, telnet (192.168.1.91,23).
  -->3. HOMER.example.com, telnet (192.168.1.90,23).
```

1. This example ends the session on MARGE. The current session is still HOMER. You can close any other session without affecting the status of the current session.

```
TELNET>CLOSE 2
%TELNET-S-LCLCLOSED, Local connection closed
-TELNET-I-SESSION, Session 02, host marge.example.com, port 23
%TELNET-I-CURRSESSION, current session is now 3, homer.example.com
```

2. This example ends the current session on HOMER and defaults to the session on BART. Because you are closing the current session, the Telnet client resets the current session to the “next” connected session.

```
TELNET>CLOSE
%TELNET-S-LCLCLOSED, Local connection closed
-TELNET-I-SESSION, Session 03, host homer.example.com, port 23
%TELNET-I-CURRSESSION, current session is now 1, bart.example.com
```

DEFINE/KEY

Associates an equivalence string and a set of attributes with a key on the terminal keyboard.

Format

`DEFINE/KEY` *`key-name`* `["]`*`equivalence-string`*`["]`

Parameters

`key-name`

Name of the key to define. The below table lists key designations for three terminal types:

- On LK201 terminals, the numeric keypad, editing keypad (except the $ and ^ arrow keys), or function key row (except F1 through F5).
- On VT52 terminals, all definable keys are on the numeric keypad.
- On VT100-type terminals, you can also define the ⇐ and ⇒ keys. On VT200 terminals, the ⇐, ⇒, and F6 through F14 keys are for command line editing. Issue the DCL command `SET TERMINAL/ NOLINE_EDITING` to define these keys before you run the Telnet client. You can also press `Ctrl+V` to enable keys F7 through F14.

Key designations for three terminal types:

Key Name	LK201	VT100-type	VT52
PF1	PF1	PF1	[blue]
PF2	PF2	PF2	[red]
PF3	PF3	PF3	[gray]
PF4	PF4	PF4	n/a
KP0,...KP9	0,...,9	0,...,9	0,...,9
PERIOD	.	.	.
COMMA	,	,	,

MINUS	-	-	-
ENTER	Enter	ENTER	ENTER
LEFT	‹	‹	‹
RIGHT	fi	fi	fi
Find (E1)	Find		
Insert_Here (E2)	Insert_Here		
Remove (E3)	Remove		
Select (E4)	Select		
Prev_Screen (E5)	Prev_Screen		
Next_Screen (E6)	Next_Screen		
HELP	Help		
DO	Do		
F6,...,F20	F6,...,F20		

equivalence-string
String to substitute when you press the key. If the string contains spaces, enclose it in quotes.

Qualifiers

/ECHO
/NOECHO
/ECHO (the default) displays the equivalence string on your screen after you press the key. /NOECHO disables this. Use /NOECHO with /TERMINATE only.

/IF_STATE=(state-name[,state-name,...])
/NOIF_STATE

/IF_STATE specifies one or more *state-names* (alphanumeric strings separated by commas) for the key definition to be in effect. You can omit the parentheses if you specify only one *state-name*. /NOIF_STATE is the default, where the current state applies.

Establish states using the /SET_STATE qualifier (see below). If you specify several *state-names*, you can define a key to have the same function in all the specified states.

/LOCK_STATE
/NOLOCK_STATE
/LOCK_STATE specifies that the state set by the /SET_STATE qualifier remains in effect until explicitly changed. /NOLOCK_STATE is the default, where the state set by /SET_STATE is in effect only for the next definable key that you press or for the next read terminating character that you type.

You can only specify /LOCK_STATE with /SET_STATE.

/SET_STATE=*state-name*
/NOSET_STATE
/SET_STATE specifies the *state-name* (an alphanumeric string) to set when pressing the key. *state-name* is an alphanumeric string. The default is /NOSET_STATE, where the current locked state, if any, remains in effect.

/TERMINATE
/NOTERMINATE
Specifies whether to terminate (execute) the current equivalence string when you press the key. /NOTERMINATE (the default) lets you create key definitions that insert text into command lines, at prompts, or into other text you type.

EXIT

Exits the Telnet client utility and returns to the DCL level. If there is an open connection or log file, the Telnet client closes it before exiting. Once you exit, all connections to remote hosts are disconnected.

Format

```
EXIT
```

Synonyms

```
QUIT
BYE
Ctrl+Z
```

FLUSH

Discards all characters currently in the output stream from the server. Ignored if no connection is open.

> **Note:** Unlike the flush character, the `FLUSH` command does not use the timing-mark option.

Format

```
FLUSH
```

HELP

Obtains help on using the Telnet client utility. TELNET help uses the OpenVMS interactive help facility.

To exit the help facility, press the **RETURN** key until you return to the `TELNET>` prompt.

Format

`HELP [topic]`

Parameter

topic
Topic on which you want help. Optional.

OPEN

Opens a connection to a remote host. You can open up to ten connections at any one time. The connection remains open until you log out of the remote host, or use the `CLOSE` or `EXIT` command at the `TELNET>` prompt.

> **Note:** The same parameters and qualifiers apply to the TELNET command on the DCL level as apply to the `OPEN` command within TELNET.

Format

`OPEN [host [port]]`

Synonyms

`CONNECT [host [port]]`
`SET HOST [host [port]]`

Parameters

host

Name of the remote host to which you want to connect. The host must exist on the network.

Enter `OPEN host` to open a remote connection and start the login sequence, if any. If you omit *host* and a connection is open, the Telnet client resumes the session to that host.

port

Nonstandard service name or number of the remote port to which you want to connect. The default is `TELNET` or `23` (for the TELNET server). Use only to connect to a nonstandard server. As an alternative, use the `/PORT` qualifier (DO NOT use both in the same command).

Qualifiers

/CREATE *[*=(PERMANENT, BROKE_TIMO=*seconds*, CLOSE_DASSGN, INTERVAL=*seconds*, *[*NO*]*KEEPALIVE, NOOPCOM, NOTCONNECTED_OK, RETRIES=*number*), SHUT_ABORT) *]*

Associates the local client end of the TELNET connection to an NTA device. Lets you use the connection for terminal activities such as printing or running applications. Supports /RAW, /LOGICAL, and /TIMEOUT.

The /CREATE keyword creates the NTA device as pre-allocated so that it is not deleted when exiting TELNET. However, deallocating the device deletes it automatically when there are no process channels assigned to it (the reference count drops to zero). The PERMANENT keyword causes the client NTA device NOT to be deleted automatically when there are no process channels assigned to it, thus creating a permanent connection similar to an application LTA device for LAT. As with LAT, if the TELNET connection is broken, the Telnet client device tries to reconnect to the specified host and port. Further parameters control the broken connection and reconnection algorithms:

BROKE_TIMO=*seconds*	Used to determine when a connection is broken. (Note that the OPEN /TIMEOUT qualifier value is used in establishing the connection, and another timeout of eight minutes is used when sending data.) If omitted, the /TIMEOUT value is used. Also applies to non-permanent NTA devices (when using OPEN/CREATE without the PERMANENT keyword).
CLOSE_DASSGN	Specifies that when the last channel is de-assigned from the NTA device, the underlying TCP connection is closed. The default is NOT to close the TCP connection. Use with the PERMANENT keyword only.
INTERVAL=*seconds*	Connection retry interval, the minimum time to wait until another connect is attempted. The default is 120 seconds (two minutes). Use with the PERMANENT keyword only.
KEEPALIVE or NOKEEPALIVE	Controls whether keep-alive segments are sent to the remote port. The default is KEEPALIVE. Also applies to non-permanent NTA devices (when using OPEN/CREATE without the PERMANENT keyword).
NOOPCOM	Specifies that no OPCOM messages are used when a permanent NTA device fails to reconnect or reconnects after an initial failure. OPCOM messages are sent by default.

NOTCONNECTED_OK	A permanent NTA device is created even if a TCP connection cannot initially be set up.
RETRIES=*number*	Number of times to try to reconnect after a connection breaks; the default is -1, handled as an unsigned number and thus actually 4,294,967,295, which is, in effect, infinite. Use with the PERMANENT keyword only.
SHUT_ABORT	Specifies that a permanent NTA device will do extra TCP device clean up after the underlying TCP connection is shutdown. This is similar to doing NETCU> KILL CONNECTION for a closed TCP device.

Setting RETRIES to 0 means that when either end closes the TCP connection, no reconnects automatically occur. However, a reconnection attempt is made without delay when a write operation to the permanent NTA device occurs. If RETRIES is not set to 0, automatic retries occur when the connection closes. If all those retries fail, and a write is done later to the NTA device, then the specified number of retries is attempted.

Here is a typical command to create a TELNET connection to a printer (note the use of /RAW to avoid sending TELNET options negotiation data):

```
$ TELNET /RAW /CREATE=(PERM, RETRIES=0, CLOSE) host port
```

After TELNET creates a permanent NTA device with an underlying TCP connection, the NTA device's reference count drops to 0; thus, the TCP connection is closed. When a write operation occurs to the NTA device, an attempt is made to re-establish the TCP connection. Meanwhile the data being written is held so that it can be sent when reconnected. If all reconnects fail, the write data is dropped. When the application de-assigns its channels to the NTA device, its TCP connection is again closed.

To specify that the permanent NT device should be treated as a local terminal rather than a remote terminal (to allow for spooling of the device), add the LOCAL keyword to the TELNET /CREATE qualifier:

```
$ TELNET /CREATE=(PERM,LOCAL)
```

/LOGICAL=*name* [/TABLE=*table*][/MODE=*mode*]
Logical name defined for the allocated NTA device. Use only with the /CREATE qualifier. The *table* values are PROCESS (the default), JOB**,** GROUP**,** or SYSTEM. The *mode* values are SUPERVISOR (the default) or EXECUTIVE.

/PORT=*port*

Nonstandard service name or number of the remote port to which you want to connect. The default is `23` (for the TELNET server). Use only to connect to a nonstandard server. Alternatively, use the *`port`* command parameter (DO NOT use both in the same command).

`/RAW`
Specifies a raw, binary connection that does not adhere to the TELNET protocol. Use only with the `/CREATE` qualifier.

`/TIMEOUT=`*`seconds`*
Timeout time for establishing the TELNET control connection. If not specified, the default value of `120` seconds (2 minutes) applies. The minimum allowable value is `20`.

Examples

1. This example creates a permanent NTA device for the connection to MARGE port 7 for the user application. In case the connection goes down, it is set up so that automatic reconnection retries occur every 10 seconds for a total of 10 retries.

```
TELNET>OPEN /LOGICAL=MY_PORT -
_TELNET>/CREATE=(PERMANENT,INTERVAL=10,RETRIES=10) MARGE 7

%TCPWARE_TELNET-I-CREATED, _NTA2: created

$ @MY_APPLICATION MY_PORT
```

2. This example displays the results of incorrectly using the *`port`* parameter value (telnet) together with the `/PORT` qualifier and value in a single command:

```
TELNET>OPEN DAISY TELNET /PORT=23

%TCPWARE_TELNET-W-CONFLICT illegal combination of command elements - check
documentation
```

RESUME

Resumes the current connection if you do not specify a session number. If you specify a session number, resumes the connection associated with the session number, as displayed by the SHOW STATUS (or STATUS) command.

Format

RESUME [*session-number*]

Parameter

session-number

Session number to resume, based on the session number the SHOW STATUS command displays. If omitted, resumes the current connection.

Examples

1. This example resumes the session on BART. The Telnet client does not display a message if the user resumes the current session:

```
TELNET>SHOW STATUS
Connected session:
-->1. BART.example.com, telnet (192.168.1.92,23).

TELNET>RESUME
(bart)$
```

2. This example resumes session 2 on MARGE:

```
TELNET>STATUS
Connected sessions:
   1. BART.example.com,  telnet (192.166.1.92,23).
   2. MARGE.example.com, telnet (192.166.1.91,23).
-->3. HOMER.example.com, telnet (192.162.1.90,23).

TELNET>RESUME 2

%TCPWARE_TELNET-I-RESUME, resuming session 2, MARGE.example.com
(marge)$
```

SEND

Sends TELNET control functions or option negotiations to a remote host.

Format

```
SEND {control-function | {command option}}
```

Parameters

control-function

The below table lists the available TELNET control functions. Send a control function to gain access to functions of the remote host that are not available from the keyboard.

Control Function	Definition
AO	Abort Output
AYT	Are You There
BACKWARD	Sends the current Telnet client Backward character
BRK	Break
EC	Erase Character
EL	Erase Line
ESCAPE	Sends the current The Telnet client Escape character
FORWARD	Sends the current The Telnet client Forward character
GA	Go-Ahead
IP	Interrupt Process
NOIP	Do Not Interrupt Process

SYNCH	SYNCH signal

Command

One of the following TELNET protocol commands used in options negotiation:

DO	WILL	DONT	WONT

Option

Negotiated TELNET option. The Telnet client supports the following *option* keywords:

ECHO	for the ECHO option SEND WILL ECHO is an invalid command. The Telnet client does not allow the user to send this option negotiation to the TELNET server.
BINARY, or TRANSMIT_BINARY	for the TRANSMIT-BINARY option
SGA, or SUPPRESS_GO_AHEAD	for the SUPPRESS-GO-AHEAD option

SET *[*NO*]*AO

Defines, changes, or disables the "abort output" (AO) character. During a TELNET session, if you enter the defined AO character, the Telnet client sends the TELNET AO control function to the server instead of the actual character.

Format

```
SET AO char
SET NOAO
```

Parameter

char

When entered, this character sends the TELNET AO control function to the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default AO character. Define the initial AO character using the `TCPWARE_TELNET_AO` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_AO 15`
- `$ DEFINE/PROCESS TCPWARE_TELNET_AO """^O"""`

Both commands set the AO character to `Ctrl+O` (ASCII 15). They are equivalent.

Qualifiers

/FLUSH
/NOFLUSH

If you specify `/FLUSH`, the Telnet client discards all characters currently in the output stream from the server when sending the AO control function. The Telnet client uses the TELNET timing-mark option to accomplish this (the server does not have to support this option for this feature to work). If you specify `/NOFLUSH`, the Telnet client sends only the AO control function. If you omit both, the previous setting remains. The initial default is `/FLUSH`.

If there is no response to the timing-mark option, the Telnet client may continue to discard output from the server. Use the `FLUSH` command to resume normal operation.

/SYNCH
/NOSYNCH
Sends the AO command followed by the SYNCH signal.

Examples

1. Each of these equivalent commands sets the AO character to `Ctrl+O` (ASCII 15):

```
TELNET>SET AO "^O"
TELNET>SET AO 15
```

2. This example removes the previous character definition, if any, for the AO control function:

```
TELNET> SET NOAO
```

SET *[*NO*]*AYT

Defines, changes, or disables the "are you there" (AYT) character. If you enter the defined AYT character during a TELNET session, the Telnet client sends the TELNET AYT control function to the server instead of the actual character.

Format

```
SET AYT char
SET NOAYT
```

Parameter

char

When entered, this character sends the TELNET AYT control function to the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default AYT character. Define the initial AYT character using the `TCPWARE_TELNET_AYT` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_AYT 7`
- `$ DEFINE/PROCESS TCPWARE_TELNET_AYT """^G"""`

Both commands set the AYT character to `Ctrl+G` (ASCII 7). They are equivalent.

Qualifiers

/SYNCH

Sends the AYT command followed by the SYNCH signal.

Examples

1. Each of these equivalent commands sets the AYT character to `Ctrl+G` (ASCII 7):

```
TELNET>SET AYT "^G"
TELNET>SET AYT 7
```

2. This example removes the previous character definition, if any, for the AYT control function:

```
TELNET>SET NOAYT
```

SET *[NO]*BACKWARD

Defines, changes, or disables the "backward (one session)" (BACKWARD) character. If you enter the BACKWARD character during a TELNET session, the "previous" numbered session becomes active. The previous numbered session is the session with the next lowest session number than the current session.

If the current session already has the lowest session number, the session with the highest session number becomes active. If there is only one active session available, that session remains active. In this case `SET BACKWARD` has no effect.

Format

```
SET BACKWARD char
SET NOBACKWARD
```

Parameter

char

When entered, this character causes the "previous" numbered session to become active. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default BACKWARD character. Define the initial BACKWARD character using the `TCPWARE_TELNET_BACKWARD` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_BACKWARD 2`
- `$ DEFINE/PROCESS TCPWARE_TELNET_BACKWARD """^B"""`

Both commands set the BACKWARD character to `Ctrl+B` (ASCII 2). They are equivalent.

Examples

1. Each of these equivalent commands sets the BACKWARD character to `Ctrl+B` (ASCII 2):

```
TELNET>SET BACKWARD "^B"
```

```
TELNET>SET BACKWARD 2
```

2. This example removes the previous character definition, if any, for the BACKWARD control function:

```
TELNET>SET NOBACKWARD
```

SET *[*NO*]*BINARY

Initiates negotiations to enable the TRANSMIT BINARY option for the client and server. This command:

- Pertains only to the current session.
- Automatically resumes the current session.

Use the `SET NOBINARY` command to initiate negotiations to disable the TRANSMIT BINARY option for the client and server.

Format

```
SET BINARY
SET NOBINARY
```

SET [NO]BRK

Defines, changes, or disables the break (BRK) character. If you define the BRK character during a TELNET session, the Telnet client sends the TELNET BRK control function to the server instead of the actual character.

Format

```
SET BRK char
SET NOBRK
```

Parameter

char

When entered, this character sends the TELNET break control function to the server. Specified in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default BRK character. Define the initial BRK character using the TCPWARE_TELNET_BRK logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_BRK 29`
- `$ DEFINE/PROCESS TCPWARE_TELNET_BRK """^]"""`

Both commands set the break character to **Ctrl/]** (ASCII 29). They are equivalent.

Qualifiers

/FLUSH
/NOFLUSH

If you specify /FLUSH, the Telnet client discards all characters currently in the output stream from the server when sending the BRK function. The Telnet client uses the TELNET timing-mark option to accomplish this (the server does not have to support this option for this feature to work).

If you specify `/NOFLUSH`, the Telnet client sends only the BRK function. If you omit both, the previous setting remains. The initial default is `/FLUSH`.

> **Note:** If a server fails to respond properly to the timing-mark option, the Telnet client may continue to discard output from the server. In this case, use the `FLUSH` command to resume normal operation.

Examples

1. Each of these equivalent commands sets the break character to **Ctrl+]** (ASCII 29):

```
TELNET>SET BRK "^]"
TELNET>SET BRK 29
```

2. This example removes the previous character definition, if any, for the break control function:

```
TELNET>SET NOBRK
```

SET DEBUG

Enables or disables the display of debugging information.

Format

`SET DEBUG /CLASS=[()keyword[,…)]`

Qualifier

/CLASS[=*keyword*]

`SET DEBUG` requires the `/CLASS` qualifier. The optional *`keyword`* specifies the classes of debugging information to enable or disable. Use parentheses for multiple keywords separated by commas. The below table lists the supported keywords.

Keyword	Description
`ALL`	Enables the display of all classes.
`OPTIONS`	Enables the display of options negotiation information. The Telnet client displays messages when it sends or receives TELNET options.
`NONE`	Disables the display of all classes.

The initial setting is `NONE`.

`SET DEBUG` alone, or `SET DEBUG /CLASS` without the keyword, shows the current debug classes.

Examples

1. This example enables the display of options negotiation information:

```
TELNET>SET DEBUG/CLASS=OPTIONS
```

SET DELETE_ALLOWED

Allows deletion of an NTA device originally set up as permanent. The deletion occurs when there are no process channels assigned to the device.

See the `OPEN /CREATE` command for details on creating permanent NTA devices.

Format

`SET DELETE_ALLOWED` *`nta-device`*

Parameter

`nta-device`
NTA device set up using `OPEN /CREATE=(PERMANENT…)`.

Example

This example allows the NTA33: device to be deleted when no channels are assigned to it:

```
TELNET>SET DELETE NTA33:
```

SET [NO]EC

Defines, changes, or disables the "erase character" (EC) character. If you enter the defined EC character during a TELNET session, the Telnet client sends the TELNET EC control function to the server instead of the actual character.

Format

```
SET EC char
SET [NO]EC
```

Parameter

char

When entered, this character sends the TELNET EC control function to the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default EC character. Define the initial EC character using the `TCPWARE_TELNET_EC` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_EC 4`
- `$ DEFINE/PROCESS TCPWARE_TELNET_EC """^D"""`

Both commands set the EC character to **`Ctrl/D`** (ASCII 4). They are equivalent.

Examples

1. Each of these equivalent commands sets the EC character to Ctrl+D (ASCII 4):

```
TELNET>SET EC "^D"
TELNET>SET EC 4
```

2. This example removes the previous character definition, if any, for the EC control function:

```
TELNET>SET NOEC
```

SET [NO]EL

Defines, changes, or disables the "erase line" (EL) character. If you enter the defined EL character during a TELNET session, The Telnet client sends the TELNET EL control function to the server instead of the actual character.

Format

```
SET EL char
SET NOEL
```

Parameter

char

When entered, this character sends the TELNET EL control function to the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default EL character. Define the initial EL character using the `TCPWARE_TELNET_EL` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_EL 21`
- `$ DEFINE/PROCESS TCPWARE_TELNET_EL """^U"""`

Both commands set the EL character to **`Ctrl/U`** (ASCII 21). They are equivalent.

Examples

1. Each of these equivalent commands sets the EL character to Ctrl+U (ASCII 21):

```
TELNET>SET EL "^U"
TELNET>SET EL 21
```

2. This example removes the previous character definition, if any, for the EL control function:

```
TELNET>SET NOEL
```

SET [NO]ESCAPE

`SET ESCAPE` changes the escape (attention) character. This command allows you to change the character to a key that is more convenient. The default escape character is `Ctrl+\`. You may want to change the escape character if the remote host uses that character to perform some function or if your terminal cannot generate the character.

`SET NOESCAPE` disables the escape (attention) character.

Format

```
SET ESCAPE char
SET NOESCAPE
```

Synonym

```
ESCAPE = SET ESCAPE
```

Parameter

char

You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

You can redefine the default escape (attention) character by defining the logical `TCPWARE_TELNET_ESCAPE` (in the process, job, group, or system logical name tables). The logical value has the same syntax as *`char`*. To define it, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_ESCAPE 24`
- `$ DEFINE/PROCESS TCPWARE_TELNET_ESCAPE """^X"""`

Both commands set the escape character to ASCII code 24 (`Ctrl+X`). They are equivalent.

- `$ DEFINE/SYSTEM/EXEC TCPWARE_TELNET_ESCAPE -1`

The `-1` value disables the escape (attention) character.

Examples

1. Each of these equivalent commands sets the escape character to `Ctrl+X` (ASCII 24):

```
TELNET>SET ESCAPE "^X"
TELNET>SET ESCAPE 24
```

2. This example sets the escape character to right brace (`}`):

```
TELNET>SET ESCAPE "}"
```

3. This example removes the previous escape (attention) character definition, if any:

```
TELNET>SET NOESCAPE
```

SET [NO]FLUSH

Defines, changes, or disables the flush character.

If you enter the defined flush character during a TELNET session, the Telnet client discards all characters currently in the output stream from the server. The Telnet client uses the TELNET timing-mark option to accomplish this (a TELNET server need not support this option for this feature to work).

> **Note:** If a server fails to respond properly to the timing-mark option, the Telnet client may continue to discard all output from the server. In this case, use the `FLUSH` command to resume normal operation.

Format

```
SET FLUSH char
SET NOFLUSH
```

Parameter

char

When entered, this character discards all characters currently in the output stream from the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default flush character. Define the initial flush character using the `TCPWARE_TELNET_FLUSH` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_FLUSH 15`
- `$ DEFINE/PROCESS TCPWARE_TELNET_FLUSH """^O"""`

Both commands set the flush character to `Ctrl+O` (ASCII 15). They are equivalent.

Examples

1. Each of these equivalent commands sets the flush character to `Ctrl+O` (ASCII 15):

```
TELNET>SET FLUSH "^O"
TELNET>SET FLUSH 15
```

2. Removes the previous character definition, if any, for the flush feature.

```
TELNET>SET NOFLUSH
```

SET [NO]FORWARD

Defines, changes, or disables the "forward [one session]" (FORWARD) character. If you enter the defined FORWARD character during a TELNET session, the "next" numbered session becomes active. The next numbered session is the session with the next highest session number than the current session.

If the current session already has the highest session number, the session with the lowest session number becomes active. If there is only one active session available, that session remains active. In this case `SET FORWARD` has no effect.

Format

```
SET FORWARD char
SET NOFORWARD
```

Parameter

char

When entered, this character causes the "next" numbered session to become active. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default FORWARD character. Define the initial FORWARD character using the `TCPWARE_TELNET_FORWARD` logical name (in the process, job, group, or system logical name tables). This logical value has the same syntax as char. To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_FORWARD 1`
- `$ DEFINE/PROCESS TCPWARE_TELNET_FORWARD """^A"""`

Both commands set the FORWARD character to `Ctrl+A` (ASCII 1). They are equivalent.

Examples

1. Each of these equivalent commands sets the FORWARD character to `Ctrl+A` (ASCII 1):

```
TELNET>SET FORWARD "^A"
TELNET>SET FORWARD 1
```

2. This example removes the previous character definition, if any, for the FORWARD control function:

```
TELNET>SET NOFORWARD
```

SET *[*NO*]*GA

Defines, changes, or disables the "go-ahead" (GA) character. If you enter the defined GA character during a TELNET session, the Telnet client sends the TELNET GA control function to the server instead of the actual character.

Format

```
SET GA char
SET NOGA
```

Parameter

char

When entered, this character sends the TELNET GA control function to the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default GA character. Define the initial GA character using the `TCPWARE_TELNET_GA` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_GA 9`
- `$ DEFINE/PROCESS TCPWARE_TELNET_GA """^I"""`

Both commands set the GA character to `Ctrl+A` (ASCII 9). They are equivalent.

Examples

1. Each of these equivalent commands sets the GA character to `Ctrl+A` (ASCII 9):

```
TELNET>SET GA "^I"
TELNET>SET GA 9
```

2. This example removes the previous character definition, if any, for the GA control function:

```
TELNET>SET NOGA
```

SET [NO]IP

Defines, changes, or disables the "interrupt process" (IP) character. If you enter the defined IP character during a TELNET session, the Telnet client sends the TELNET IP control function to the server instead of the actual character.

Format

```
SET IP char
SET NOIP
```

Parameter

char

When entered, this character sends the TELNET IP control function to the server. You can specify this character in either of the following formats:

- Numeric: ASCII value of the character
- String: Character string enclosed in quotes. Specify control characters by typing a caret (^) before the character.

There is no default IP character. Define the initial IP character using the `TCPWARE_TELNET_IP` logical name (in the process, job, group, or system logical name tables). To define the logical, use one of the following formats:

- `$ DEFINE/PROCESS TCPWARE_TELNET_IP 25`
- `$ DEFINE/PROCESS TCPWARE_TELNET_IP """^Y"""`

Both commands set the IP character to `Ctrl+Y` (ASCII 25). They are equivalent.

Qualifiers

/FLUSH
/NOFLUSH

With `/FLUSH`, the Telnet client discards all characters currently in the server's output stream when sending the IP control function. It uses the TELNET timing-mark option (the server does not have to support this option for this feature to work). With `/NOFLUSH`, the Telnet client sends only the IP control function. If you omit both, the previous setting remains. The initial default is `/FLUSH`.

If a server fails to respond properly to the timing-mark option, the Telnet client can continue to discard all output from the server. If so, use `FLUSH` to resume normal operation.

/SYNCH
Sends the IP command followed by the SYNCH signal.

Examples

1. Each of these equivalent commands sets the IP character to `Ctrl+Y` (ASCII 25):

```
TELNET>SET IP "^Y"
TELNET>SET IP 25
```

2. This example removes the previous character definition, if any, for the IP control function.

```
TELNET>SET NOIP
```

SET [NO]LOCAL_FLOW_CONTROL

Controls the handling of the XON/XOFF characters (`Ctrl+S` and `Ctrl+Q`) when connected to a remote system. `Ctrl+S` stops transmission and `Ctrl+Q` resumes TELNET transmission. Under normal conditions, the terminal driver processes `Ctrl+S` and `Ctrl+Q` locally and does not send them to the remote TELNET server.

The Telnet client supports RFC 1372 (*Telnet Remote Flow Control Option*), which lets the remote server tell the client when to enable and disable local flow control. These commands are not related to that option, but rather let the user control the local flow control setting if the remote server does NOT support the Remote Flow Control Option.

Use `SET NOLOCAL_FLOW_CONTROL` to pass the `Ctrl+S` and `Ctrl+Q` characters to the remote TELNET server and NOT process them 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.

Format

```
SET LOCAL_FLOW_CONTROL
SET NOLOCAL_FLOW_CONTROL
```

Example

```
TELNET>SET NOLOCAL
TELNET>SHOW STATUS
Telnet client  V6.1  Copyright (c) Process Software

Connected session: -->1. beans.example.edu, telnet (192.168.0.50)

Terminal type:  VT300
Local flow control: OFF
"^D" is the escape (attention) character.
```

SET LOG

Opens or closes a log file. The Telnet client uses a log file to save the output from a remote host. While connected to a remote host, the Telnet client also puts all output the remote host sends your terminal into the log file.

`SET LOG` logs output from every connected session. If multiple connections exist, there is no way to specify that you want to log only output from a specified session to the log file.

Format

`SET LOG [`*`file`*`]`

Opens the local file *`file`* and begins logging. To close a log file (and stop logging), enter `SET LOG` with no file specification.

Parameter

`file`
OpenVMS file specification of the file that logs the remote host's output. If omitted, the Telnet client closes the present log file (if there is one).

Qualifiers

`/DATA`
`/NODATA`
`/DATA` logs all data sent to the specified file (the default). `/NODATA` disables this.

`/OPTIONS`
`/NOOPTIONS`
`/OPTIONS` prints option negotiations to the specified log file, in addition to performing normal logging. `/NOOPTIONS` (the default) disables options printing.

Examples

1. This example opens the file `TEXT.LOG` and enables logging:

```
TELNET>SET LOG TEXT.LOG
```

2. This example closes a log file and stops logging:

```
TELNET>SET LOG
```

3. This example opens the file `TEXT.LOG`, enables normal logging, and prints options negotiations to the `TEXT.LOG` file:

```
TELNET>SET LOG TEXT.LOG /OPTIONS
```

4. This example opens the file `TEXT.LOG` and prints only option negotiations (and no data) to the `TEXT.LOG` file:

```
TELNET>SET LOG TEXT.LOG /OPTIONS /NODATA
```

SET TERMINAL_TYPE

Requests the server to support a specific terminal type or types if negotiating the terminal type option.

Normally, you do not need to use this command. The telnet client uses the following default list of supported terminal types: VT52, VT55, VT61, VT62, VT100, VT102, VT125, VT131, VT132, VT200, VT220, VT240, VT300, VT320, VT340, and IBM-3278-*model-number*.

If you specify an IBM-3278 terminal type, make sure your local terminal supports the screen size associated with the specified model number. If your terminal does not support the screen size, the data will not display properly.

Use the SHOW STATUS or SHOW OPTIONS commands to show the current terminal type used.

The TCPWARE_TELNET_TERMINAL_TYPE logical performs the same function as the SET TERMINAL_TYPE command. This logical requires the following syntax:

```
$ DEFINE/SYSTEM/EXEC TCPWARE_TELNET_TERMINAL_TYPE "type"
```

Format

SET TERMINAL_TYPE *type[,type,...]*

Parameter

type

A valid terminal type. The Telnet client requests the server to support these types in the specified order.

Examples

1. This example requests the server to support the VT300 and VT100 terminal types, in that order:

```
TELNET>SET TERMINAL_TYPE VT300, VT100
```

2. This example requests the server to support the IBM-3278-3 terminal type. If possible, the Telnet client resizes the local window to accommodate a 32 x 80 screen size for model 3.

```
TELNET>SET TERMINAL_TYPE IBM-3278-3
```

SET TRANSLATION

Sets the carriage return/line feed (CR/LF) character translation.

Format

```
SET TRANSLATION
```

Qualifiers

/RECEIVE=*keyword*

Specifies the mapping for characters received from the server before they become output. See the below table for the keywords and their meaning. The default is `/RECEIVE=NONE`.

/SEND=*keyword*

Specifies the mapping for characters entered at the keyboard before the Telnet client sends them to the server. See the below table for the keywords and their meaning. The default is `/SEND=CR`.

Keyword	Translation
CR	The Telnet client translates the carriage return character to a CR/LF sequence
LF	The Telnet client translates the line feed character to a CR/LF sequence
NONE	The Telnet client does not translate characters to the CR/LF sequence

SET [NO]XDISPLOC

Enables or disables setting your current X display location on the remote end, when communicating with a remote TELNET server that also supports this option. Client TELNET checks whether the logical `DECW$DISPLAY` is defined. If it is, and if the remote server asks for the X display location, the X display server address is transmitted to the remote system.

Use `SET NOXDISPLOC` before making a connection to disable sending the X display location.

Format

```
SET XDISPLOC
SET NOXDISPLOC
```

Example

```
TELNET>SET NOXDISPLOC
TELNET>OPEN ALPHA
$ SHOW DISPLAY
Error opening DECW$DISPLAY as input
No such device available
ALPHA>
```

SHOW OPTIONS

Displays information about the options in effect.

Options modify the way TELNET handles your terminal over the network. When you first establish a connection, both hosts negotiate for the options to use based on the options that each host supports. You can also use the `SEND` command to change options.

Format

```
SHOW OPTIONS
```

Example

```
TELNET>SHOW OPTIONS
Current TELNET options status:

   Remote ECHO
   No remote TRANSMIT-BINARY (normal ASCII)
   No local TRANSMIT-BINARY (normal ASCII)
   Remote SUPPRESS-GO-AHEADS
   Local SUPPRESS-GO-AHEADS
   No remote END-OF-RECORD
   No local END-OF-RECORD
   Local TERMINAL-TYPE: VT300
   Local FLOW-CONTROL: ON
   Local WINDOW-SIZE: 80x35
   Local X-DISPLAY-LOCATION: 192.168.5.195:0.0
```

SHOW STATUS

Displays information about all open TELNET connections and your current TELNET session.

The screen displays the following information:

- Session number, name and internet address of each remote host if a connection is open. An arrow (-->) indicates the current session.
- The list of supported terminal types if no remote connection is open.
- The terminal type used, if a remote connection is open and the Telnet client negotiated for the terminal type.
- Whether local flow control is ON or OFF.
- Name of the log file if one is open.
- Name of the host character set.
- Name of the terminal character set.
- The current "abort output" (AO), "are you there" (AYT), backward, break (BRK), "erase character" (EC), "erase line" (EL), escape, forward, flush, "interrupt process" (IP), and "go-ahead" (GA) characters (if defined).

Format

```
SHOW STATUS
```

Synonym

```
STATUS
```

Example

```
TELNET> SHOW STATUS
Telnet client V6.1  Copyright (c) Process Software
Connected sessions:
    1. bart.example.com, telnet (192.168.1.92,23).
 -->2. marge.example.com, telnet (192.168.1.91,23).
"^\" is the escape (attention) character

Terminal type: IBM-3278-2
Local flow control: ON

Host Character Set: CANADIAN
Terminal Character Set: LATIN1

"^C" is the escape (attention) character.
```

SHOW TRANSLATION

Displays the current translation settings made using `SET TRANSLATION`. Both the received and sent translations appear.

Format

```
SHOW TRANSLATION
```

Example

```
TELNET> SHOW TRANSLATION
No characters are translated to CRLF when received.
CR is translated to CRLF when sent.
```

SPAWN

Executes DCL commands.

> **Note:** You cannot `SPAWN` with CAPTIVE accounts.

Format

`SPAWN [command-line]`

Synonym

`Z[command-line]`

Parameter

command-line
DCL command line that you want executed. If omitted, the Telnet client spawns an interactive subprocess. To return to TELNET from an interactive subprocess, logout of that subprocess.

Examples

1. This example displays the time on your local host without leaving the TELNET utility:

```
TELNET>SPAWN SHOW TIME
  3-Nov-2021 14:02:48
```

2. This example initiates DCL command mode and returns the DCL prompt:

```
TELNET>SPAWN
$ SHOW TIME
  3-Nov-2021 14:02:51
$ LOGOUT
  Process SMITH_1 logged out at 3-Nov-2021 14:02:54.34
TELNET>
```

To exit the DCL command mode and return to TELNET, enter the `LOGOUT` command at the DCL prompt.

13. TFTP: Trivial File Transfers

Introduction

The Trivial File Transfer (TFTP) utility provides the user interface to TFTP. This program allows a user to transfer files to and from a remote host. TFTP primarily allows remote diskless systems to read bootstrap images over the network. TFTP uses UDP to make transfers. It does not provide user login validation.

See Chapter 4 of the *Installation and Configuration Guide for information* about configuring the TFTP server.

TCPware's FTP implementation is a more complete file transfer facility than TFTP. See Chapter 3, *FTP: Transferring Files*, for details on FTP.

Invoking TFTP

To invoke TFTP, enter at the DCL prompt:

```
$ TFTP [host [port]]
```

If you specify a host name, TFTP uses that host for subsequent file transfers. If you also specify a port number, TFTP uses the specified host and port for subsequent file transfers.

Command Reference

You interact with TFTP by typing commands at the `TFTP>` prompt. The TFTP client supports the following OpenVMS-style commands:

CONNECT	MODE	REXMT	TIMEOUT	HELP
GET	PUT	STATUS	TRACE	QUIT

TFTP offers 20-line recall on the command level.

CONNECT

Sets the host and, optionally, the port number for subsequent file transfers. Note that TFTP uses UDP and, therefore, does not maintain the connection between transfers.

Format

CONNECT *host* [*port*]

Synonym

OPEN

Parameters

host

Name of the remote host to which you want to connect. The host must exist on the network.

port

Service name or number of the remote port that you want to connect to. The default port number is 69 for read and write requests. You do not need to specify the port number unless you are connecting to a nonstandard server.

Example

Each of these equivalent commands connects to host SIGMA for a file transfer:

```
tftp>connect sigma
tftp>open sigma
```

GET

Gets a file from the previously specified remote host. TFTP writes the local file as a STREAM_LF formatted file.

Since TFTP does not authenticate the client, the server allows access only to files in the directory and its subdirectories defined by the `TCPWARE_TFTP_ROOT` logical.

The server converts UNIX filenames with their directories into VMS filenames as in the below table. The directory specification is *`dir`* and the filename specification with its extension is *`filename.ext`*.

UNIX Filename...	Is Converted to VMS Filename...
`dir/filename.ext`	`[.dir]`*`filename.ext`*
/`dir/filename.ext`	`[.dir]`*`filename.ext`*

Format

`GET` *`remote-file`* `[`*`local-file`*`]`

Parameters

`remote-file`
Input file specification on the remote host.

`local-file`
Output file specification on the local host. If omitted, Client-TFTP uses the remote-file filename and extension.

Examples

This command transfers the `US-DOMAIN-INFO.TXT` file from the previously specified host:

```
tftp>get us-domain-info.txt
```

This command transfers the US-DOMAIN-INFO.TXT file from the previously specified host as file LOCALSTUFF.TXT:

```
tftp> get us-domain-info.txt localstuff.txt
```

HELP

Displays a brief help message summarizing the commands.

Format

`HELP [command]`

Parameter

command
Optional command for which you want help.

Examples

This command provides help for the `CONNECT` and `GET` commands:

```
tftp>help connect
connect to remote tftpd
tftp> help get
receive file
```

MODE

Sets the file transfer mode to type; type may be either ASCII or BINARY. The initial type is ASCII.

Format

MODE *type*

Parameter

type
The mode type, either ASCII or BINARY.

Example

This command changes the transfer mode to BINARY:

```
tftp>mode binary
tftp> status
Connected to SIRIUS.example.com.
Mode:
octet Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
```

PUT

Puts a file to the previously specified remote host.

Since TFTP does not authenticate the client, the server allows access only to files in the directory and its subdirectories defined by the TCPWARE_TFTP_ROOT logical.

The server converts OpenVMS filenames with their directories into UNIX filenames as shown below. The directory specification is *dir* and the filename specification with its extension is *filename.ext*.

UNIX Filename...	Is Converted to VMS Filename...
dir/filename.ext	[.*dir*]*filename.ext*
/dir/filename.ext	[.*dir*]*filename.ext*

Format

PUT *local-file* [*remote-file*]

Parameters

local-file
Input file specification on the local host.

remote-file
Output file specification on the remote host. If omitted, TFTP uses the *local-file* filename and extension.

Examples

This command transfers the US-DOMAIN-INFO.TXT file to the previously specified host:

```
tftp>put us-domain-info.txt
```

This command transfers the US-DOMAIN-INFO.TXT file to the previously specified host as file REMOTESTUFF.TXT:

```
tftp>put us-domain-info.txt remotestuff.txt
```

QUIT

Exits the TFTP program. You can also use **Ctrl+Z** and EXIT to exit the program.

Format

```
QUIT
```

Examples

Each of these equivalent commands exits from TFTP:

```
tftp>quit
tftp>exit
tftp>Ctrl+Z
```

REXMT

Sets the retransmit timer, in seconds. The initial value is 5 seconds.

The value you enter for REXMT is also used together with the specified maximum timeout (set using the TIMEOUT command) to determine the number of times to try and the actual maximum timeout reported in a status request (STATUS).

If the default 5 seconds retransmit interval is used together with the default 25 seconds maximum timeout, the number of times to try is 5, according to the formula:

Max-timeout = Rexmt-interval x Tries

The REXMT value you enter is always reported (unchanged) on the Rexmt-interval line in a STATUS request. However, the maximum timeout may be recalculated before being reported as Max-timeout.

See the TIMEOUT command for details on Max-timeout recalculation.

Format

REXMT [*time*]

Parameter

time
The time value to set the retransmit timer. If omitted, the value is 5 seconds.

Example

This command changes the retransmit timer (Rexmt-interval) to 10 seconds (and the subsequent STATUS command shows the result). The Max-timeout is set to five times the Rexmt-interval by default.

```
tftp> rexmt 10
tftp>status
Connected to SIRIUS.example.com.
Mode:
octet Tracing: off
Rexmt-interval: 10 seconds, Max-timeout: 50 seconds
```

STATUS

Displays the current status and parameter settings.

The `Max-timeout` reported is based on the following computation:

`Max-timeout` = `Rexmt-interval` x `Tries`

The number of tries (`Tries`) is initially 5 unless adjustments are made to the `Max-timeout` and `Rexmt-interval` values (see below for an example).

> **Note:** The total retransmission period (`Max-timeout`) value displayed may be slightly different from that set using the `TIMEOUT` command. (See the `TIMEOUT` command for an explanation.)

Format

```
STATUS
```

Examples

This command shows the connection status, file transfer mode (`Mode:`), packet trace flag status (`Tracing:`), retransmit timer (`Rexmt-interval:`), and total retransmission period (`Max-timeout:`) values over the period of several adjustments. (See the `TIMEOUT` command for an explanation of the `Max-timeout` recalculations.)

```
tftp> connect spica
tftp>stat
Connected to spica.example.com.
Mode: netascii Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp>rexmt 4
tftp>stat
Connected to spica.example.com.
Mode: netascii Tracing: off
Rexmt-interval: 4 seconds, Max-timeout: 20 seconds
tftp>timeout 40
tftp>stat
Connected to spica.example.com.
```

```
Mode: netascii Tracing: on
Rexmt-interval: 4 seconds, Max-timeout: 40 seconds
tftp>timeout 30
tftp>stat
Connected to spica.example.com.
Mode: netascii Tracing: on
Rexmt-interval: 4 seconds, Max-timeout: 28 seconds
```

TIMEOUT

Sets the total retransmission period, in seconds. The initial value is 25 seconds.

Minor adjustments to the specified retransmission period as reported using `STATUS` can occur based on concurrent changes made to the retransmit timer setting (`REXMT`). The retransmission period is calculated based on the following formula:

`Max-timeout` = `Rexmt-interval` x `Tries`

The `Tries` value must be an integer value. Thus, if the `Max-timeout` specified using the `TIMEOUT` command forms a non-integer ratio with the `Rexmt-interval` value, the `Max-timeout` is adjusted accordingly. (See the example below.)

Format

`TIMEOUT [`*`time`*`]`

Parameter

time

The total retransmission period, in seconds. If omitted, the value is 25 seconds.

Examples

Note the way in which the retransmission period is adjusted in this example:

```
tftp>connect spica
tftp>stat
Connected to spica.example.com.
Mode: netascii Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp>rexmt 4
tftp>stat
Connected to spica.example.com.
Mode: netascii Tracing: off
Rexmt-interval: 4 seconds, Max-timeout: 20 seconds
tftp>timeout 40
tftp>stat
Connected to spica.example.com.
Mode: netascii Tracing: on
Rexmt-interval: 4 seconds, Max-timeout: 40 seconds
```

```
tftp>timeout 30
tftp> stat
Connected to spica.example.com.
Mode: netascii Tracing: on
Rexmt-interval: 4 seconds, Max-timeout: 28 seconds
```

- The retransmit timer and number of tries are both set to 5 by default, so that initially the `Max-timeout` is 25.
- With the retransmit timer (`rexmt`) reset to 4, the `Max-timeout` changes to 4 x 5 = 20.
- Doubling the maximum timeout (`timeout 40`), recalculates the number of retries to 40 / 4 = 10.
- Changing the maximum timeout to 30 (with the `rexmt` still set to 4) recalculates the retries to 7 and adjusts the `Max-timeout` to 4 x 7 = 28.

TRACE

Toggles the packet trace flag.

Format

```
TRACE
```

Example

This command enables packet tracing. A `GET` operation shows a timeout on a file transfer read request.

```
tftp>trace
Packet tracing on.
tftp>status
Connected to SIRIUS.example.com.
Mode: octet Tracing: on
Rexmt-interval: 10 seconds, Max-timeout: 60 seconds
tftp>get rfc999.txt pokertwo.txt
rqst sent RRQ <file=rfc999.txt, mode=octet>
rqst sent RRQ <file=rfc999.txt, mode=octet>
rqst sent RRQ <file=rfc999.txt, mode=octet>
rqst sent RRQ <file=rfc999.txt, mode=octet>
rqst sent RRQ <file=rfc999.txt, mode=octet>
rqst sent RRQ <file=rfc999.txt, mode=octet>
Receive request timed out
```

14. WHOIS: Username Directory Services

The WHOIS utility allows Internet users to query the Network Information Center (NIC) username directory services.

To invoke WHOIS, enter at the DCL prompt:

```
$ WHOIS name
```

name is the user's name or other search keyword.

The utility tries to connect to the NIC WHOIS server (ds.internic.net) and displays any returned information.

The source code for this utility is in the TCPWARE_COMMON:[TCPWARE.EXAMPLES]WHOIS.C file.

15. Accessing Remote Systems with the Secure Shell (SSH) Utilities

The SSH implementation for TCPware provides the client software for allowing secure interactive connections to other computers.

The following topics describe how to configure, maintain, and use the TCPware client and following utilities:

- SSHKEYGEN
- SSHAGENT
- SSHADD
- CERTTOOL
- CERTVIEW
- CMPCLIENT

SSH Protocol Support

The SSH client software supports both the SSH1 and SSH2 protocols. SSH1 and SSH2 are different, and incompatible protocols. The SSH1 implementation is based on the V1.5 protocol, and the SSH2 implementation is based on the V2 protocol. While SSH2 is generally regarded to be more secure than SSH1, both protocols are offered by TCPware, and although they are incompatible, they may exist simultaneously on server systems, including TCPware servers. The SSH client identifies the protocol(s) offered by any given server. If both SSH2 and SSH1 protocols are offered, the client will always use SSH2. Otherwise, the client will use the correct protocol based on the server's capability.

TCPware's SSH2 implementation is FIPS 140-2 level 2 compliant, as determined by the Computer Security Division of the National Institute of Science and Technology (NIST). **Note that this does not apply to SSH1.**

SSH Client Return Status Codes

In versions of TCPware prior to V5.x, the return status codes from the SSH clients listed above were based on UNIX-style status codes, causing problems for many VMS users. Beginning with TCPware

Qualifier	Description
/ALLOW_REMOTE_CONNECT	Allow remote hosts to connect local port forwarding ports. The default is only localhost; may connect to locally binded ports.
/CIPHER=(*cipher-1,...,cipher-n*)	Select encryption algorithm(s).
/COMPRESS	Enable compression.
/CONFIG_FILE=*file*	Read an alternative client configuration file.
/DEBUG=*level*	Set debug level.
/ESCAPE_CHARACTER=*char*	Set escape character; "none" = disable (default: ~).
/HELP	Display help text.
/IDENTITY_FILE=*file*	Identity file for public key authentication.
/IDKEY=(*key1,key2,...,keyn*)	Specifies the key(s) to be used for public key authentication. If specified, the IDENTIFICATION file is ignored.
/IPV4	Use IPV4 protocol to connect.
/IPV6	Use IPV6 protocol to connect.
/LOCAL_FORWARD=([protocol/]*listen-port:host:port,...*)	Causes the given port on the local (client) host to be forwarded to the given host and port on the remote side. The system to which SSH connects acts as the intermediary between the two endpoint systems. Port forwardings can be specified in the configuration file. Only system can forward privileged ports. See the *Port Forwarding* section for more details.
/LOG_FILE=*logfilename*	Log all terminal activity to the specified log file. Defaults to SYS$DISK:[]SSH.LOG if *logfilename* is not specified.
/MAC=*(mac-1,...,mac-n)*	Select MAC algorithm(s).

/NO_AGENT_FORWARDING	Disable authentication agent forwarding.
/NO_X11_FORWARDING	Disable X11 connection forwarding.
/OPTION=*(option-1,...option-n)*	Gives options in the format used in the configuration file. This is useful for specifying options for which there is no separate command-line flag. The options have the same format as a line in the configuration file and are processed prior to any keywords in the configuration file. For example: /OPTION=(CompressionLevel=6)
/PORT=*port*	Connect to this port on server system. Server must be listening on the same port.
/QUIET	Quiet Mode. Causes all warning and diagnostic messages to be suppressed. Only fatal errors display.
/REMOTE_FORWARD=([protocol/]*listen-port:host:port,...*)	Forward remote port to local address. These cause SSH to listen for connections on a port, and forward them to the other side by connecting to host port.
USE_NONPRIV_PORT	Use a non-privileged (>1023) source port.
USER=*user*	Log in to the server system using this username.
VERBOSE	Display verbose debugging messages. Equal to /DEBUG=2.
/VERSION	Display version number of the client.

V5.x, a logical name may be defined that will cause the SSH clients listed above to use VMS-style return codes. If the logical name isn't defined, the old-style codes will still be used by default. Refer to Appendix C in this U*ser's Guide* for a description of the new status codes.

To enable the new status codes instead of using the pre-TCPware V5.x codes, the logical name TCPWARE_SSH_NEW_STATUS_CODES must be defined system wide.

Secure Shell Client (remote login program)

```
$ SSH hostname[#port] [qualifiers] [command]
```

or

```
$ SSH "user@hostname[#port]" [qualifiers] [command]
```

SSH (Secure Shell) is a program for logging into and executing commands on a remote system. It replaces rlogin, rsh, and telnet, and provides secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can be forwarded over the secure channel. SSH connects and logs into the specified hostname.

Initial Server System Authentication

When an initial connection is made from the client system to the server system, a preliminary authentication of the server is made by the client. To accomplish this, the server system sends its public key to the client system.

SSH maintains a directory containing the public keys for all hosts to which it has successfully connected. For each user, this is the [.SSH2.HOSTKEYS] directory off the individual SYS$LOGIN directory. In addition, a system-wide directory of known public keys exists in the system directory pointed to by the logical name TCPWARE_SSH2_HOSTKEY_DIR, and this may be populated by the system manager. Both directories are searched as needed when establishing a connection between systems. Any new host public keys are added to the user's HOSTKEYS directory. If a host's identification changes, SSH warns about this and disables password authentication to prevent a trojan horse from getting the user's password. Another purpose of this mechanism is to prevent man-in-the-middle attacks that could be used to circumvent the encryption. The SSH configuration option StrictHostKeyChecking can be used to prevent logins to a system whose host key is not known or has changed.

Host-Based Authentication

Host-based authentication relies on two things: the existence of the user's system and username in either SSH_DIR:HOSTS.EQUIV or in the individual user's SYS$LOGIN:.RHOSTS or SYS$LOGIN:.SHOSTS file; and the server system having prior knowledge of the client system's public host key.

For SSH2:

When a user logs in:

1. The server checks the SSH_DIR:HOSTS.EQUIV file, and the user's SYS$LOGIN:.RHOSTS and SYS$LOGIN:.SHOSTS files for a match for both the system and username. Wildcards are not permitted.
2. The server checks to see if it knows of the client's public host key (SSH2_DIR:HOSTKEY.PUB on VMS client systems) in either the user's SYS$LOGIN:[SSH2.KNOWNHOSTS] directory or in the system-wide directory pointed to by the TCPWARE_SSH2_KNOWNHOSTS_DIR logical name. The key file is named *FQDN_algorithm*.PUB. For example, if the client system is foo.bar.com and its key uses the DSS algorithm, the file that would contain its key on the server would be FOO_BAR_COM_SSH-DSS.PUB. This key file must exist on the server system before attempting host-based authentication.
3. If the key file is found by the server, the client sends its digitally signed public host key to the server. The server will check the signature for validity.

For SSH1

This form of authentication alone is not allowed by the server because it is not secure. The second (and primary) authentication method is the RHOSTS or HOSTS.EQUIV method combined with RSA-based host authentication. It means that if the login would be permitted by .RHOSTS, .SHOSTS, SSH_DIR:HOSTS.EQUIV, or SSH_DIR:SHOSTS.EQUIV file, and if the client's host key can be verified (see SYS$LOGIN:[.SSH]KNOWN_HOSTS and SSH_DIR:SSH_KNOWN_HOSTS), only then is login permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing.

Public Key Authentication

The SSH client supports DSA-based authentication for SSH2 sessions, and RSA-based authentication for SSH1 sessions. The scheme is based on public-key cryptography. There are cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key.

For SSH1:

SSH supports RSA-based authentication. The scheme is based on public key cryptography. There are cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key.

RSA is one such system. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key (`SYS$LOGIN:[.SSH]AUTHORIZED_KEYS` lists the public keys permitted for log in), and only the user knows the private key.

When the user logs in:

1. The SSH client program tells the server the key pair it would like to use for authentication.
2. The server checks if this key pair is permitted.
3. If it is permitted, the server sends the SSH client program running on behalf of the user a challenge (a random number) encrypted by the user's public key. The challenge can only be decrypted using the proper private key.
4. The user's client then decrypts the challenge using the private key, proving that he/she knows the private key but without disclosing it to the server.

SSH implements the RSA authentication protocol automatically.

The key identity files are created with `SSHKEYGEN`. To create the RSA key pair files with TCPware, run `SSHKEYGEN` to create the RSA key pair: `IDENTITY` and `IDENTITY.PUB`. Both of these files are stored in the user's `SYS$LOGIN:[.SSH]` directory. `IDENTITY.` is the private key; `IDENTITY.PUB` is the public key.

Once you have created your identity files:

1. Transfer the `IDENTITY.PUB` file to the remote machine.
2. Update the `AUTHORIZED_KEYS` file on the remote machine by appending the contents of the public key file to the `SYS$LOGIN:[.SSH]AUTHORIZED_KEYS` file on the remote host. The format of the `AUTHORIZED_KEYS` file requires that each entry consists of a single long line.

After this, the user can log in without giving the password. RSA authentication is much more secure than rhosts authentication. The most convenient way to use RSA authentication may be with an authentication agent.

For SSH2:

When the user logs in:

1. The client reads possible keys to be used for authentication from its `IDENTIFICATION` file. Note that this file does not contain the actual keys; rather, it contains the name of the key files.
2. The client sends to the server its list of keys.

3. The server compares each key that it received to see if it can match this key with one of those specified in the AUTHORIZATION file.
4. The server tells the client the key that was accepted. The client then "signs" the key with a digital signature that only the server with the proper key could verify and sends the signature to the server.
5. The server verifies the signature.

Password Authentication

The password is sent to the remote host for checking. The password cannot be seen on the network because all communications are encrypted. When the server accepts the user's identity it either executes the given command or logs into the system and gives the user a normal shell on the remote system. All communication with the remote command or shell will be encrypted automatically.

Using Publickey Authentication with SSH

When a parameter such as a username or hostname is quoted, it's always passed verbatim to the other side. When it's not quoted, it's lowercased. The username entered is used when constructing the digital signature for a key.

On the host side, the uppercase username will be used, and on the server side, the lowercased username (the default on the server since VMS isn't case-sensitive) will be used to generate the digital signature of the public key that's being used, as shown in the following examples:

```
$ SSH2 "XXXXXXX@HOSTNAME" command
XXXXXXX is the username that was specified in all uppercase letters.
Public key authentication fails.
$ SSH2 "xxxxxxx@HOSTNAME command
xxxxxx is the username that was specified in all lowercase letters.
Public key authentication is successful.
```

Break-in and Intrusion Detection

Care must be exercised when configuring the client to minimize problems due to intrusion records created by OpenVMS security auditing. The SSH user should consult the system manager to determine the authentication methods offered by the SSH server. Examples of such authentication methods include HostBased, PublicKey, and Password. The client should be configured to not attempt any authentication method that is not offered by the server.

If a client attempts authentication methods not offered by the server, the OpenVMS security auditing system may log several intrusion records for each attempt to create a session to that server. The result being that the user could be locked out and prevented from accessing the server system without intervention from the server's system manager.

Session Termination

The user can disconnect with "~ .". All forwarded connections can be listed with "~#". All available escapes can be listed with "~?". A single tilde character can be sent as "~~" (or by following the tilde with a character other than those described above). The escape character must always follow a carriage return to be interpreted as special. The escape character "~" can be changed in configuration files or on the command line.

The session terminates when the command or shell on the remote system exits, or when the user logs out of an interactive session, and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of SSH.

X11 Forwarding

With X11 in use, the connection to the X11 display forwards to the remote side any X11 programs started from the interactive session (or command) through the encrypted channel. Also, the connection to the real X server is made from the local system. The user should not set `DECW$DISPLAY` manually. Forwarding of X11 connections can be configured on the command line or in configuration files.

The `DECW$DISPLAY` value set by SSH points to the server system with a display number greater than zero. This is normal and happens because SSH creates a "proxy" X server on the server system for forwarding the connections over the encrypted channel.

SSH sets up "fake" Xauthority data on the OpenVMS server, as OpenVMS does not support Xauthority currently. It generates a random authorization cookie, stores it in Xauthority on the server, and verifies that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server system (and no cookies are sent in plain text).

Configuring the SSH Client

The SSH client uses only SSH2 configuration keywords. There are no SSH1-specific configuration keywords for the SSH client.

The SSH client obtains configuration data from the following sources (in this order):

1. Command line options.
2. User's configuration file (`SYS$LOGIN [.SSH2]SSH2_CONFIG`). See the below table for details.
3. System-wide configuration file (`SSH2_DIR:SSH2_CONFIG`).

For each parameter, the first obtained value is used. The configuration files contain sections bracketed by `Host` specifications. That section applies only for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line. Since the first obtained value for each parameter is used, more host-specific declarations should be given near the beginning of the file, and general defaults at the end.

Keyword	Value	Default	Description
`AllowedAuthentications`	List	All methods except for `hostbased`	Permitted techniques, listed in desired order of attempt. These can be the following: `keyboard-interactive`, `password`, `publickey`, `kerberos-1@ssh.com`, `kerberos-tgt-1@ssh.com`, `kerberos-2@ssh.com`, `kerberos-tgt-2@ssh.com`, and `hostbased`. Each specifies an authentication method. The authentication methods are tried in the order in which they are specified with this configuration parameter.
`AuthenticationSuccessMsg`	Y/N	Y	Print message on successful authentication
`AuthorizationFile`	Filename	Authorization	Authorization file for public key authentication. See below for more information on the contents of this file.
`BatchMode`	Y/N	N	Don’t prompt for any input during session
`Ciphers`	Cipher list	None	Supported encryption ciphers
`ClearAllForwardings`	Y/N	N	Ignore any specified forwardings

Compression	Y/N	N	Enable data compression
DebugLogFile	Filename	None	Specify the file to hold debug information. If used with the QuietMode keyword turned on as well, only the first part of the log information will be written to SYS$ERROR, until the DebugLogFile keyword is parsed. If QuietMode is not used, all debug output will go to both SYS$ERROR and the log file.
DefaultDomain	Domain		Specify domain name
EscapeChar	Character	"~"	Set escape character (^=ctrl key)
ForwardAgent	Y/N	Y	Enable agent forwarding
ForwardX11	Y/N	Y	Enable X11 forwarding
GatewayPorts	Y/N	N	Allow connection to locally forwarded ports
Host	Pattern		Begin the per-host configuration section for the specified host
HostCA	Certificate	None	Specifies the CA certificate (in binary or PEM (base64) format) to be used when authenticating remote hosts. The certificate received from the host must be issued by the specified CA and must contain a correct alternate name of type DNS (FQDN). If the remote hostname is not fully qualified, the domain specified by configuration option DefaultDomain is not fully

			qualified, the domain specified by configuration option `DefaultDomain` is appended to it before comparing it to certificate alternate names. If no CA certificates are specified in the configuration file, the protocol tries to do key exchange with ordinary public keys. Otherwise, certificates are preferred. Multiple CAs are permitted.
`HostCANoCRLs`	Certificate	None	Similar to `HostCA`, but disables CRL checking for the given ca-certificate.
`IdentityFile`	Filename	Identification	Name of identification file for public key authentication
`KeepAlive`	Y/N	Y	Send keepalives
`LdapServers`	ServerURL	None	Specified as `ldap://`*`server.domainname`*`:389` CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be checked if the point exists. Otherwise, the comma-separated server list given by option `LdapServers` is used. If intermediate CA certificates are needed in certificate validity checking, this option must be used or retrieving the certificates will fail.

LocalForward	Port, Socket		Local port forwarding
Macs	Algorithm	None	Select MAC (Message Authentication Code) algorithm
NoDelay	Y/N	N	Disable Nagle (TCP_NODELAY)
NumberOfPasswordPrompts	Number	3	Number of times the user is prompted for a password before the connection is dropped
PasswordPrompt	String	"%U's password:"	Password prompt. The following substitutions may be made within the prompt string: %U = insert users's username %H = insert user's system name
Port	Port	22	Server port number
QuietMode	Y/N	Y	Quiet mode - only fatal errors are displayed
RandomSeedFile	Filename	Random_seed	Random seed file
RekeyIntervalSeconds	Seconds	3600	Number of seconds between doing key exchanges during a session. 0 = disable
RemoteForward	Port, Socket		Remote port forwarding
SendNOOPPackets	Y/N		Send NOOP packets through the connection. Used typically to

			prevent a firewall from closing an interactive session
`StrictHostKeyChecking`	Y/N/Ask	Y	Behavior on host key mismatch
`TryEmptyPassword`	Y/N	N	Attempt an empty password first when doing password authentication. **Note:** Doing so may result in an extra intrusion being logged.
`User`	Username		Remote username
`VerboseMode`	Y/N	N	Verbose mode
`VerifyHostKeyDNS`	Y/N/ASK	N	Determines if the host key fingerprint must be matched in DNS.

The user may specify default configuration options, called “stanzas”, for different destination systems in their `SSH2_CONFIG` file. The format of this within the configuration file is:

```
hostname:
  keyword        value
  keyword        value
hostname2:
   keyword       value
   keyword       value
```

For example:

```
petunia:
  port          17300
  user          dilbert
  host          petunia.example.com
rose:
  port          16003
  user          dogbert
  host          rose.example.com
  allowedauthentications password
```

```
*.beans.com:
  user            limabean
  keepalive       no
  ciphers         3des,twofish
```

In the preceding example:

- When a user types **SSH PETUNIA**, the client will connect to port 17300 on petunia.example.com, and will use the default username of "dilbert".
- When a user types **SSH ROSE**, the client will connect to port 16003 on host rose.example.com, and will use the default username of "dogbert", and only allow password authentication.
- When a user types **SSH *anything*.BEANS.COM**, the client will use the default username of "limabean", will not send keepalives, and will only allow 3DES or TWOFISH encryption.

The user may override defaults specified in configurations. Options that are specified on the command line override any like options in the configuration file. For example, if the user wants to use a username of "catbert" when connecting to host rose instead of the default username of "dogbert", this would be specified as:

```
$ SSH /USER=CATBERT ROSE
```

Authorization File Options

The authorization file has the same general syntax as the configuration files. The following keywords may be used.

Key
This is followed by the filename of a public key in the `[.SSH2]` directory file that is used for identification when contacting the host. If there is more than one key, they are all acceptable for login.

Options
This keyword, if used, must follow the "Key" keyword above. The various options are specified as a comma-separated list. See below for documentation of the options.

Command
This keyword is deprecated (though it still works). Use Options instead.

Options that can be specified:

allow-from
deny-from

Specifies that in addition to public-key authentication, the canonical name of the remote host must match the pattern(s). These parameters follow the logic of {`Allow,Deny`} Hosts described in detail in `sshd2_config`. Specify one pattern per keyword, and multiple keywords can be used.

command="*command*"
This is used to specify a "forced command" that will be executed on the server side instead of anything else when the user is authenticated. This option might be useful for restricting certain public keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Notice that the client may specify TCP/IP and/or X11 forwarding unless they are explicitly prohibited.

idle-timeout=*time*
Sets idle timeout limit to time in seconds (`s` or nothing after number), in minutes (`m`), in hours (`h`), in days (`d`), or in weeks (`w`). If the connections have been idle (all channels) for that long a period of time, the connection is closed down.

no-port-forwarding
Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, for example, in connection with the command option.

no-x11-forwarding
Forbids X11 forwarding when this key is used for authentication. An X11 forward request by the client will return an error.

SSH Client/Server Authentication Configuration Examples

Host-based Authentication Example

The following is an example of how to set up the SSH client and SSH2 server for host-based authentication:

```
$!
$! First, generate the host key - ONLY if it doesn't exist!
$!
$ netcu sshkeygen /ssh2 /host
Generating 1024-bit dsa key pair
4 oOo.oOo.oOo
```

```
Key generated.
1024-bit dsa, myname@myclient.foo.com, Thu MAR 04 2021 13:43:54
Private key saved to tcpware_ssh2_hostkey_dir:hostkey.
Public key saved to tcpware_ssh2_hostkey_dir:hostkey.pub

$ directory tcpware_ssh2_hostkey_dir:hostkey.*

Directory TCPWARE_SPECIFIC:[TCPWARE.SSH2.HOSTKEYS]

HOSTKEY.;1          HOSTKEY.PUB;1

Total of 2 files
$!
$! Copy the client system public key to the user directory on the
$! server
$!
$! DECnet must be running before you execute the following
$! commands:
$!
$ copy tcpware_ssh2_hostkey_dir:hostkey.pub -
_$ myserv"myname myuser"::[.ssh2.knownhosts]myclient_foo_com_ssh-
dss.pub
$!
$! Finally, log into the server system and ensure the
$! SSH_DIR:HOSTS.EQUIV file is correct
$!
$ SET HOST MYSERV

      Welcome to OpenVMS (TM) VAX Operating System, Version V7.3
Username: myname
Password: ********
      Welcome to OpenVMS VAX V7.3

    Last interactive login on Monday,  1-MAR-2021 17:07
    Last non-interactive login on Monday, 1-MAR-2021 08:30

MYSERV_$ type tcpware:hosts.equiv
#
# HOSTS.EQUIV - names of hosts to have default "r" utility access
# to the local # system.
#
#   This file should list the full domain-style names.
#
#   This list augments the users' SYS$LOGIN:.RHOSTS file for
#   authentication.
#   Both the .RHOSTS and the HOSTS.EQUIV files are cached by
#   TCPWARE_ - see the section entitled "RLOGIN and RSHELL
#   Authentication Cache" in the _Administrator's Guide_ for more
#   information on controlling the cache.
```

```
#
#   This file is ignored for the users SYSTEM and ROOT. SYSTEM and
#   ROOT must have a SYS$LOGIN:.RHOSTS file if you want to use
#   RSHELL or RLOGIN with them.
#
localhost
myclient.foo.com       myname
MYSERV_$
MYSERV_$ logout
  MYNAME      logged out at 1-MAR-2021 13:46:58.91
%REM-S-END, control returned to node MYCLIENT::
```

Public Key Authentication Example

The following is an example of how to set up the SSH client and SSH2 server for public key authentication:

```
$!
$! First, generate a key tuple
$!
$ netcu sshkeygen /ssh2
Generating 1024-bit dsa key pair
   1 oOo.oOo.oOo.

Key generated.
1024-bit dsa, myname@myclient.foo.com, Thu Mar 04 2021 14:06:10
Passphrase :
Again      :
Private key saved to DISK$USERDISK:[MYNAME.SSH2]id_dsa_1024_a.
Public key saved to DISK$USERDISK:[MYNAME.SSH2]id_dsa_1024_a.pub
$ directory [.ssh2]id*.*/since = TODAY

Directory DKA0:[MYNAME.SSH2]

ID_DSA_1024_A.;1    ID_DSA_1024_A.PUB;1

Total of 2 files.
$!
$! Now create the IDENTIFICATION. file.  This contains the name of
$! all the keys you wish to use for public-key authentication.
$!
$ set default [.ssh2]
$ copy tt: identification.
  idkey id_dsa_1024_a
  ^Z
$!
$! Copy the key to the user's [.ssh2] directory on the server
$! system
$!
$ copy id_dsa_1024_a.pub myserv"myname mypass"::[.ssh2]
```

```
$!
$! Now log into the server system and create the AUTHORIZATION
$! file
$!
$ set host myserv

      Welcome to OpenVMS (TM) VAX Operating System, Version V7.3

Username: myname
Password: *********
      Welcome to OpenVMS VAX V7.3

    Last interactive login on Tuesday,  2-MAR-2021 13:46
    Last non-interactive login on Tuesday,  2-MAR-2021 13:47

$ set default [.ssh2]
$ directory [.ssh2]id*.*

Directory DKA0:[MYNAME.SSH2]

ID_DSA_1024_A.PUB;1

Total of 1 file.
$ copy tt: authorization.
key id_dsa_1024_a.pub
^Z
$ logout
  MYNAME      logged out at 2-MAR-2021 14:10:26.16
%REM-S-END, control returned to node MYCLIENT::
```

The public key assistant and its subsystem can also be used to transfer public keys and maintain the authorization file for implementations that support the public key subsystem.

SSH1 Example

```
$ ! An example of the procedure of setting up SSH to enable
$ ! RSA-based authentication.
$ ! Using SSH client node to connect to an SSH server node.
$ !
$ ! On the client node
$ !
$ NETCU SSHKEYGEN /SSH1
Initializing random number generator...
Generating p:  ..............................++ (distance 662)
Generating q:  ................++ (distance 370)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key
(DISK$SYS_LOGIN:[MYNAME.ssh]identity.):
```

```
Enter passphrase:
Enter the same passphrase again:
Your identification has been saved in
DISK$SYS_LOGIN:[MYNAME.ssh]identity..
Your public key is:
1024 33 13428.........29361 MYNAME@long.example.com
Your public key has been saved in DISK$SYS_LOGIN:[MYNAME.ssh]identity.pub
$ !
$ ! A TCP/IP stack must be loaded on the remote system.
$ !
$  FTP DAISY /USER=MYNAME/PASSWORD=DEMONSOFSTUPIDITY -
_$ PUT DISK$SYS_LOGIN:[MYNAME.ssh]identity.PUB -
_$ DISK$SYS_LOGIN:[MYNAME.ssh]identity.PUB
long.example.com TCPware FTP user process V6.1
Connection opened (Assuming 8-bit connections)
<daisy.example.com TCPware FTP Server Process V6.1 at Thu 6-Mar-2021 3:20PM-
EDT
[Attempting to log in as myname]
<User MYNAME logged into DISK$SYS_LOGIN:[MYNAME] at Thu 6-MAR-2021 3:21PM-
EDT, job 20e00297.
<VMS Store of DISK$SYS_LOGIN:[MYNAME.SSH]IDENTITY.PUB; started.
<Transfer completed.  395 (8) bytes transferred.
<QUIT command received. Goodbye.
$
$ TELNET DAISY
Trying... Connected to DAISY.EXAMPLE.COM.


        Authorized Users Only (TM) VAX Operating System, Version V7.1

Username: MYNAME
Password: *********
     Welcome to OpenVMS (TM) VAX Operating System, Version V7.1 on node
DAISY
    Last interactive login on Thursday,  4-MAR-2021 08:07
    Last non-interactive login on Thursday,  6-MAR-2021 15:21
         Logged into DAISY at  4-MAR-2021 15:22:43.68
$ !
$ ! For the first entry into the AUTHORIZED_KEYS file copy
$ ! (or rename) the file [.SSH]IDENTITY.PUB to
$ ! [.SSH]AUTHORIZED_KEYS.
$ !
$ COPY [.SSH]IDENTITY.PUB [.SSH]AUTHORIZED_KEYS.
$
$ ! FOR SUBSEQUENT ENTRIES use the APPEND command
$ !
$ APPEND [.SSH]IDENTITY.PUB [.SSH]AUTHORIZED_KEYS.
$
$ ! A sanity check of the file protections shows
$ !
```

```
$ DIRECTORY/PROTECTION [.SSH]*.*

Directory DISK$SYS_LOGIN:[MYNAME.SSH]

AUTHORIZED_KEYS.;1    (RWE,RWED,RE,E)
IDENTITY.;1           (RWD,RWD,,)
IDENTITY.PUB;1        (RWE,RWED,RE,E)
KNOWN_HOSTS.;1        (RWD,RWD,,)
RANDOM_SEED.;1        (RWD,RWD,,)

Total of 5 files.
$ !
$ DIRECTORY/PROTECTION SSH.DIR

Directory DISK$SYS_LOGIN:[MYNAME]

SSH.DIR;1             (RWD,RWD,,)

Total of 1 file.
```

SSH2 User Authentication Using Certificates:

Client setup:

1. Copy the private key and certificate (`.crt`) into the user's `[.ssh2]` directory, and edit the `[.ssh2]` identification file, adding entry `certkey` *`private_key_name`*.

```
$ dir [.ssh2]

Directory DKA0:[DILBERT.SSH2]

AUTHORIZATION.;13 IDENTIFICATION.;1 MYCERT.;1 MYCERT1.CRT.;2

Total of 4 files.

$ type [.ssh2]identification.
  certkey mycert1
$
```

Server setup:

1. Copy the CA certificate into your `SSH2_DIR:` directory.
2. Add the following entries in `SSH2_DIR:SSHD2_CONFIG`:

```
Pki SSH2_DIR:CAcertname
Mapfile SSH2_DIR:CAcertname.map
```

The `Pki` keyword begins an authority block for a given CA certificate. There might be more than one CA certificate along with its own mapping file.

The `Mapfile` keyword specifies the location of the certificate to username mapping file.

In addition, for testing, you might use `PkiDisableCRLs yes` to disable CRL checking for the given authorization block.

3. Create the mapping file `SSH2_DIR:`*`CAcertname`*`.map` The mapping file consists of rows of the following format:

```
userid mappingrule mapdata
```

`userid` is the user ID that's allowed to login for the given cert (there might be multiple user IDs for a given certificate).

`Mappingrule` is one of `subject`, `email`, `serialandissuer` or `emailregex`.

- `Subject` means that the following *`mapdata`* is matched against the subject of the certificate.
- `Email` is the e-mail alternative subject extension (`emailregex` allows the use of regular expressions - e.g., `%subst% emailregex ([a-z}|+)@example\.com` would allow any trusted certificate having an e-mail alternative name of *`username`*`@example.com` to login with user ID *`username`*).
- `SerialAndIssuer` is the serial number and DN of the issuer separated by whitespace. DNs are used in reverse LDAP order (e.g., `c=US,o=Foobar,cn=Dilbert Dogbert`).

SSH2 Host Key Authentication Using Certificates

Server setup:

1. Create a certificate for the server. Host certificate must contain the fully qualified domain name as the DNS alternative name.
2. Copy the private key and certificate into `TCPWARE_SSH2_HOSTKEY_DIR` directory.
3. Add the following entries into the `ssh2_dir:sshd2_config` file

```
HostKeyFile tcpware_ssh_hostkey_dir:hostcert
HostCertificateFile tcpware_ssh_hostkey_dir:hostcert.crt
```

Client setup:

1. Copy the CA certificate into the `TCPWARE_SSH2_HOSTKEY_DIR` directory.
2. Add the following entries into `ssh2_dir:ssh2_config`:

```
HostCA tcpware_ssh_hostkey_dir:CAcert.crt
DefaultDomain fully_qualified_domain_of_client
```

> **Note:** For testing purposes, you can use `HostCANoCRLs` instead of `HostCA` to disable CRL checking.

Host Key Verification Using DNS

TCPware SSH can be configured to calculate the fingerprint of the host key it receives, then perform a lookup in DNS for that fingerprint. This can help prevent man-in-the-middle attacks. See RFC 4255 for more details. Refer to Appendix D of the *TCPware Management Guide* for information on configuring DNSSEC on TCPware systems.

To do this, the following conditions must be met:

- DNSSEC must be enabled and configured for the DNS used by the client.
- A host key SSHFP record must be generated, signed by the zone key, and added to the DNSSEC configuration as a type SSHFP record for the server system. The TCPware `SSH-KEYGEN2` utility can be used to display the host key SSHFP type record for DNS.
- The client configuration keyword `VerifyHostKeyDNS` must be set to `Y` or `ASK`.
- The client configuration keyword `StrictHostKeyChecking` must be set to `Y` or `ASK`.

When the host key is received from the server, and after the client goes through its normal host key checking (e.g., does the client already know about this host key), it checks the status of the `VerifyHostKeyDNS` keyword. If not set to `N`, the client calculates the fingerprint of the host key, then performs a DNS lookup of the key,

If no records are found, the user may be given the option of proceeding (if `VerifyHostKeyDNS` is set to `ASK`). If the user responds `N`, then the session is terminated. Otherwise, the host key is accepted and the session continues.

If one or more records are found, the fingerprint and type of the host key received are compared against those found in DNS. If no matches are made, the user may be given the opportunity to ignore this state (see above).

If a match was made, the `RRSET_VALIDATED` flag returned by DNSSEC is examined to see if the signing of the records can be fully trusted. If this is true, the host key processing is complete. If this flag is false, the user may be given the opportunity to ignore this state (see above).

Port Forwarding

Port forwarding is a mechanism whereby programs that use known TCP/IP ports can have encrypted data forwarded over unsecure connections. This is also known as "tunneling".

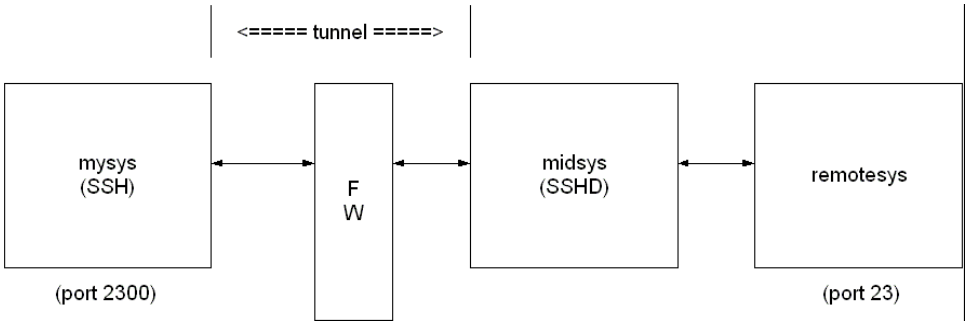
If the user is using an authentication agent, the connection to the agent is forwarded automatically to the remote side unless disabled on the command line or in a configuration file. Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either on the command line or in a configuration file.

> **Note:** Forwarded ports (tunnels) exist only as long as the SSH session that established them exists; if the SSH session goes away, so do the tunnels.

`/LOCAL_FORWARD=(`*`localport`*`:`*`remotehost`*`:`*`remoteport`*`)`
This causes *`localport`* on the system the client is running on to be forwarded to *`remotehost`*`:`*`remoteport`*. The system to which SSH2 connects acts as the intermediary between the two endpoint systems.

For example: Use port forwarding to allow a system (midsys) to encrypt and forward TELNET sessions between itself (mysys) that's outside a corporate firewall to a system (remotesys) that is inside a

corporate firewall. Note that the use of port 2300 in the examples is arbitrary.



From the DCL prompt on mysys:

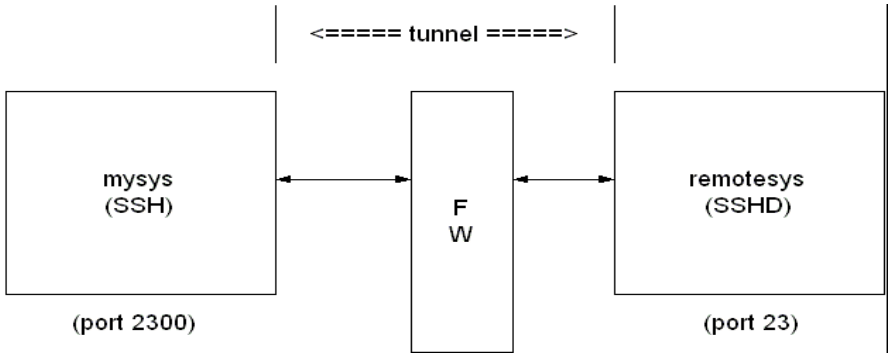
```
$ SSH midsys /local_forward=(2300:remotesys:23)
```

With the SSH session to midsys now active, type in another window on mysys:

```
$ telnet localhost /port=2300
```

This causes a connection to mysys:2300. The SSH2 client has bound to this port, and will see the connection request. SSH sends an "open channel" request to midsys, telling it there's a connect request for port 23 on remotesys. Midsys will connect to remotesys:23, and send back the port information to mysys. Mysys completes the connection request, and the TELNET session between mysys and remotesys is now in place, using the tunnel just created through the firewall between mysys and midsys.

All traffic between mysys and midsys (through the firewall) is encrypted/decrypted by SSH on mysys and SSHD on midsys, and hence, is safe. TELNET does not know this, of course, and does not care. Note that ports can also be forwarded from a localhost to the remote host that's running SSHD, as illustrated in this figure.



In this example, port 2300 on mysys is being forwarded to remotesys:23. To do this, use SSH on mysys:

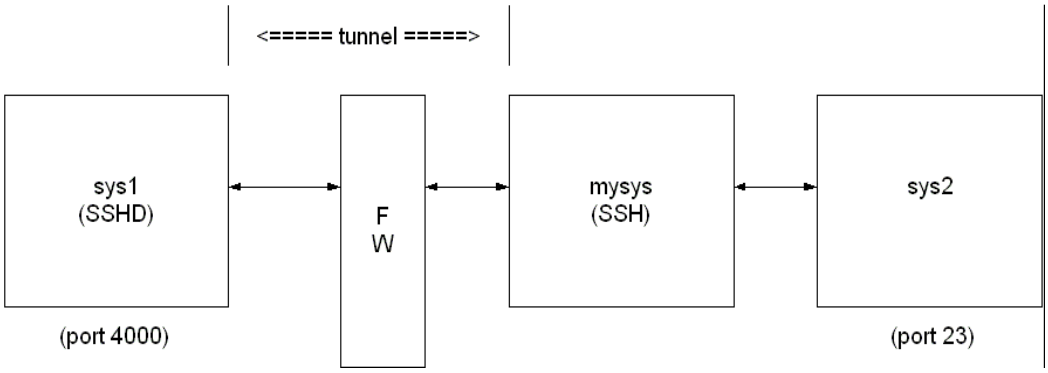
```
$ SSH remotesys /local_forward=(2300:remotesys:23)
```

Then, also on mysys, type:

```
$ telnet localhost /port=2300
```

When SSH and SSHD start their dialog, SSHD on remotesys connects back to itself, port 23, and the TELNET session is established.

/REMOTE_FORWARD=(*remoteport1*:*remotehost*:*remoteport2*)
This causes *remoteport1* on the system to which SSH connects to be forwarded to *remotehost*:*remoteport2*. In this case, the system on which the client is running becomes the intermediary between the other two systems.



For example, a user wants to use mysys to create a tunnel between sys1:4000 and sys2:23, so that TELNET sessions that originate on sys1:4000 get tunneled to sys2 through the firewall. On mysys:

```
$ SSH sys1 /remote_forward=(4000:sys2:23)
```

Now, on sys1, a user could establish a TELNET session to sys1 by doing:

```
$ telnet localhost /port=4000
```

The mechanism used for making the TELNET connection (setting up the tunnel) is essentially the same as described in the /LOCAL_FORWARD example above, except that the roles of SSH and SSHD in the dialog are reversed.

Other Files

The files in the below table are used by SSH. Note that these files generally reside in the [.SSH2] subdirectory from the user's SYS$LOGIN directory. The [.SSH2] subdirectory is created automatically on your local system the first time SSH is executed, and on a remote OpenVMS system the first time an SSH connection is made to that system. File protection for SYS$LOGIN:SSH2.DIR should be (S:RWD, O:RWD, G:, W:).

File Name	Resides On	Description
[.SSH2]SSH2_CONFIG.	Client System	This is the individual configuration file. This file is used by the SSH2 client. It does not contain sensitive information. The recommended file protection is (S:RWD,O:RWD,G:,W:).
[.SSH2]IDENTIFICATION	Client System	Contains the information about private keys that can be used for public-key authentication when logging in.
[.SSH2]ID_*alg_bits_seq*	Client System	Contains a private key for authentication. • *alg* is either RSA or DSA • *bits* is the length of the key • *seq* is an incrementing alphabetic value Thus, a key named ID_DSA_1024_A. indicates this is a private DSA key 1024 bits long, and it is the first time the key was generated using SSHKEYGEN. A user may have multiple private key files in a directory.
[.SSH2]ID_*alg_bits_seq*.PUB	Client System and Server System	Contains a public key for authentication. • *alg* is either RSA or DSA • *bits* is the length of the key • *seq* is an incrementing alphabetic value Thus, a key named ID_DSA_1024_B.PUB indicates this is a public DSA key 1024 bits long, and it is the second time the key was

		generated using SSHKEYGEN. A user may have multiple public key files in a directory.
[.SSH2.HOSTKEYS]xxx.PUB	Client System	Contains public host keys for all hosts the user has logged into. The files specifications have the format KEY_*port*_*hostname*.PUB • *port* is the port over which the connection was made • *hostname* is the hostname of the key's host. For example, if tulip.example.com was accessed via port 22, the key file would be KEY_22_TULIP_EXAMPLE_COM.PUB. If this file changes on the host (for example, the system manager regenerates the host key), SSH2 will note this and ask if you want the new key saved. This helps prevent man-in-the-middle attacks.
[.SSH2]RANDOM_SEED.	Client System	Seeds the random number generator. This file contains sensitive data and MUST have a protection of no more than (S:RWD,O:RWD,G:,W:), and it must be owned by the user. This file is created the first time the program is run and is updated automatically. The user should never need to read or modify this file. On OpenVMS systems, multiple versions of this file will be created; however, all older versions of the file may be safely purged. Use the DCL command: SET FILE /VERSION_LIMIT=*n* RANDOM_SEED to set a limit on the

		maximum number of versions of this file that may exist at any given time.
SSH_DIR:.RHOSTS	Server System	Is used in host-based authentication to list the host/user pairs that are permitted to log in. Each line of the file contains a host name (in the fully qualified form returned by name servers), and then a user name on that host, separated by a space. This file must be owned by the user and must not have write permissions for anyone else. The recommended permission is read/write for the user, and not accessible by others.
SSH_DIR:.SHOSTS	Server System	Is used the same way as .RHOSTS.
TCPWARE:HOSTS.EQUIV	Server System	Is used during .rhosts authentication. It contains fully-qualified hosts names, one per line. If the client host is found in this file, login is permitted provided client and server user names are the same. Additionally, successful RSA host authentication is required. This file should only be writable by SYSTEM.
TCPWARE:SHOSTS.EQUIV	Server System	Is processed exactly as SSH_DIR:HOSTS.EQUIV. This file may be useful to permit logins using SSH but not using rshell/rlogin.
SSH2_DIR:SSH2_CONFIG	Client System	This is a system-wide client configuration file. This file provides defaults for those values that are not specified in a user's configuration file, and for users who do not have a configuration file. This file must be world readable.

TCPWARE_SSH2_KNOWNHOSTS_DIR	Server System	Contains public host keys for all hosts the system has logged into. The files specifications have the format KEY_*port_hostname*.PUB • *port* is the port over which the connection was made • *hostname* is the hostname of the key's host. For example, if tulip.example.com was accessed via port 22, the key file would be KEY_22_TULIP_EXAMPLE_COM.PUB. If this file changes on the host (for example, the system manager regenerates the host key), SSH will note this and ask if you want the new key saved. This helps prevent man-in-the-middle attacks.

SSHKEYGEN

Generates authentication key pairs. The format of the keys is incompatible between SSH1 and SSH2. Therefore, the correct format keys must be generated for each version of the protocol to be supported.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, you need to generate a new key and copy the corresponding public key to other systems.

Each key may be protected via a passphrase, or it may be left empty. Good passphrases are 10-30 characters long and are not simple sentences or otherwise easily guessable. Note that the passphrase can be changed later, but a lost passphrase cannot be recovered, as a “one-way” encryption algorithm is used to encrypt the passphrase.

> **Note:** The host key has no password.

SSH1

```
NETCU SSHKEYGEN /SSH1 [/BITS=n] [/IDENTITY_FILE=file]
                         [/PASSPHRASE=passphrase] [/COMMENT=comment]
NETCU SSHKEYGEN /SSH1 /CHANGE_PASSPHRASE [/PASSPHRASE=old_passphrase]
                         [/NEW_PASSPHRASE=new_passphrase]
NETCU SSHKEYGEN /SSH1 /CHANGE_COMMENT [/PASSPHRASE=passphrase]
                         [/COMMENT=comment]
NETCU SSHKEYGEN /SSH1 /CHANGE_CIPHER [/IDENTITY_FILE=file]
                         [/PASSPHRASE=passphrase]
NETCU SSHKEYGEN /SSH1 /HOST [/BITS=n][/COMMENT=comment]
```

Option	Description
/BITS=*nnn*	Specify key strength in bits (default = 1024).
/CHANGE_PASSPHRASE	Change the passphrase of private key file.
/CHANGE_COMMENT	Change the comment for a key.

/CHANGE_CIPHER	Change the cipher to current default (3DES).
/COMMENT="*comment*"	Provide the comment.
/HOST	Generate the host key.
/IDENTITY_FILE=*file*	Specify the name of the host key file.
/PASSPHRASE=*ppp*	Provide the current passphrase.
/NEW_PASSPHRASE=*ppp*	Provide new passphrase.
/VERSION	Print sshkeygen version number.

SSH2

```
NETCU SSHKEYGEN /SSH2[/BITS=n][/COMMENT=comment][/KEYTYPE=type]
                    [/KEYS=(key1...keyn)]
                    [/PASSPHRASE=ppp|/NOPASSPHRASE][/STIR=file][/QUIET]
NETCU SSHKEYGEN /SSH2/HOST
                    [/BITS=n][/COMMENT=comment][/STIR=file][/QUIET]
NETCU SSHKEYGEN /SSH2/DERIVE_KEY=file
NETCU SSHKEYGEN /SSH2/EDIT=file
NETCU SSHKEYGEN /SSH2/FINGERPRINT=file
NETCU SSHKEYGEN /SSH2/INFO=file [/BASE=n]
NETCU SSHKEYGEN /SSH2/SSH1_CONVERT=file
NETCU SSHKEYGEN /SSH2/X509_CONVERT=file
NETCU SSHKEYGEN /SSH2/PKCS_CONVERT=file
NETCU SSHKEYGEN /SSH2/EXTRACT_CERTS=file
NETCU SSHKEYGEN /SSH2/HELP
NETCU SSHKEYGEN /SSH2/VERSION
NETCU SSHKEYGEN /SSH2/NOWARN
NETCU  SSHKEYGEN /SSH2/DNS_DIGEST
```

Option	**Description**
/BASE=*nnn*	Number base for displaying key info
/BITS=*nnn*	Specify key strength in bits (default = 1024).

/COMMENTS="*comment*"	Provide the comment.
/PKCS_CONVERT=*file*	Convert a PKCS 12 file to an SSH2 format certificate and private key.
/SSH1_CONVERT=*file*	Convert SSH1 identity to SSH2 format.
/X509_CONVERT=*file*	Convert private key from X.509 format to SSH2 format.
/DERIVE_KEY=*file*	Derive the private key given in *file* to public key.
/DNS_DIGEST	Calculate and display a DNSSEC SSHFP record of the local host key that can be added to a DNS configuration file.
/EDIT=*file*	Edit the comment/passphrase of the key.
/EXTRACT_CERTS=*file*	Extract certificates from a PKCS 7 file.
/FINGERPRINT=*file*	Dump the fingerprint of *file*.
/INFO=*file*	Load and display information for *file*.
/HELP	Print help text.
/HOST	Generate the host key.
/KEYS=(*key1,...,keyn*)	Generate the specified key file(s).
/KEYTYPE=(*dsa* | *rsa*)	Choose the key type: dsa or rsa.
/OPENSSH_CONVERT=*file*	Convert the specified OpenSSH key to SSH2 format
/OUTPUT_FILE=*file*	Write the key to the specified output file
/PASSPHRASE=*ppp*	Provide the current passphrase.
/NOPASSPHRASE	Assume an empty passphrase.

/QUIET	Suppress the progress indicator.
/STIR=file	Stir data from file to random pool.
/VERSION	Print sshkeygen version number.
/[NO]WARN	Enable or disable warnings if the process of generating host keys using /HOST will cause existing host keys to be overwritten. If enabled, the user will be prompted to overwrite them. If disabled, no warnings or prompts are issued if the host keys exist. Default is /WARN.

There is also a comment field in the public key file that is for the convenience to the user to help identify the key. The comment can tell what the key is for, or whatever is useful. The comment is initialized to *nnn*-bit dsa, *username*@*hostname*, *ddd mm-dd-yyyy hh*:*mm*:*ss* when the key is created unless the /COMMENT qualifier is used, and may be changed later using the /EDIT qualifier.

> **Note:** When the /HOST qualifier is used, the /KEYS=(*key1*,…*keyn*) qualifier is ignored.

> **Note:** The public key file must be world readable.

SSHAGENT (Authentication Agent)

SSHAGENT is a program that holds authentication private keys. Both SSH1 and SSH2 keys are supported by SSHAGENT. SSHAGENT may be started in the beginning of a login session by including the commands to start it in, for example, LOGIN.COM. It may also be started interactively at any time during a login session.

To start SSHAGENT, one of the three methods may be used:

1. Start it in a separate window:

```
$ SSHAGENT
```

2. Spawn it as a subprocess:

```
$ SPAWN/NOWAIT SSHAGENT
```

3. Run it in a detached process:

```
$ RUN/DETACHED/OUTPUT=AGENT.OUT/INPUT=NLA0:/PROCESS_NAME="SSH
AGENT" SSH_EXE:SSH-AGENT2
```

The agent is used for public key authentication when logging to other systems using SSH. A connection to the agent is available to all programs run by all instances of the user on a specific system. The name of the mailbox used for communicating with the agent is stored in the TCPWARE_SSH_AGENT_*username* logical name. Note that while the agent mailbox is accessible only by the user that starts the agent, a user with sufficient VMS privileges could access the agent mailbox and steal or modify keys currently loaded into the agent (although, the keys as stored on disk cannot be modified simply by accessing the agent).

The agent does not have any private keys initially. Keys are added using SSHADD. When executed without arguments, SSHADD adds the user's identity files. If the identity has a passphrase, SSHADD asks for the passphrase. It then sends the identity to the agent. Several identities can be stored in the agent; the agent can use any of these identities automatically.

The command SSHADD /LIST displays the identities currently held by the agent. The idea is that the agent is run on the user's workstation.

FILES

SYS$LOGIN:[SSH]IDENTITY

Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key. That passphrase is used to encrypt the private part of this file. This file is not used by `SSHAGENT` but is added to the agent using `SSHADD` at login.

SSHADD

Adds identities to SSHAGENT, the authentication agent. When run without arguments, SSHADD adds the file [.SSH]IDENTITY. Alternative file names can be given on the command line. If any file requires a passphrase, SSHADD asks for the passphrase from the user.

The authentication agent must be running and must have been executed by the user for SSHADD to work.

Format

SSHADD [*OPTIONS*] [*FILE*[,*FILE*,*FILE*]]

FILE is an identity or certificate file. If no file is specified, the files in the user's [.SSH2] directory are used.

Options

/HELP	Display help text.
/LIST	List all identities currently represented by the agent.
/LOCK	Lock the agent with a password.
/NOSSH1	Agent cannot use SSH1 keys.
/PURGE	Remove all identities from the agent.
/REMOVE	Remove the identity from the agent. In order to remove identities, you must either issue the command from the subdirectory that the identities are located in, or issue the command using the full path name of the identity (as is seen in an SSHADD /LIST command).
/TIMEOUT=*n*	Agent should delete this key after the timeout value (in minutes) expires.
/UNLOCK	Unlock the locked agent.

/URL	Give key to the agent as a URL.

Files

These files exist in SYS$LOGIN:

[.SSH]IDENTITY	Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key. That passphrase is used to encrypt the private part of this file. This is the default file added by SSHADD when no other files have been specified. If SSHADD needs a passphrase, it reads the passphrase from the current terminal if it was run from a terminal. If SSHADD does not have a terminal associated with it but DECW$DISPLAY is set, it opens an X11 window to read the passphrase.
[.SSH]IDENTITY.PUB	Contains the public key for authentication. The contents of this file should be added to [.SSH]AUTHORIZED_KEYS on all systems where you want to log in using RSA authentication. There is no need to keep the contents of this file secret.
[.SSH]RANDOM_SEED	Seeds the random number generator. This file should not be readable by anyone but the user. This file is created the first time the program is run and is updated every time SSHKEYGEN is run.

CERTTOOL

The CERTTOOL utility is used for different needs concerning X.509 certificates.

Format

```
certtool [options] /pk10 /subject=subject /key_usage=flags
           /extended_key_usage=flags
certview [options] /pk12 /input_files=objects
```

Valid Options

/BITS=*n*	Key strength in bits (default 2048)
/DEBUG=*n*	Set debug level to *n*
/EXTENDED_KEY_USAGE =(flag1...flagn)	(PKCS#10 only) Extended key usage flags, as a comma-separated list. Valid values are: • anyExtendedKeyUsage • ServerAuth • clientAuth • codeSigning • emailprotection No extended flags are set by default.
/HELP[=(PK10,PK12)]	Display help. More detailed help on manipulating PKCS#10 and PKCS#12 certs is available by adding the PK10 and PK12 qualifier, respectively, to the HELP switch.
/INPUT_FILES=*(file1...filen)*	(PKCS#12 only)

	List of files to include in the PFX package.
/KEY_TYPE=*type*	Create a new key of type DSA or RSA.
/KEY_USAGE=*(flag1...flagn)*	(PKCS#10 only) Key usage flags, as a comma-separated list. Valid values are: • digitalSignature • nonRepudiation • keyEncipherment • dataEncipherment • keyAgreement • keyCertSign • CRLSign • encipherOnly • decipherOnly Default values are digitalSignature and keyEncipherment.
/OPTION=(*x*,*y*)	Set certificate option *x* to *y*. The options that can be set are dependent upon the type of certificate (PKCS#10 or PKCS#12) being affected. For PKCS#10: • DNS - set certificate DNS names • Email - set certificate email addresses For PKCS#12: • KeyPBE - set the PBE scheme for shrouding keys. default means pbeWithSHAAnd3-KeyTripleDES-CBC. • SafePBE - set the PBE scheme for protecting safes. default means pbeWithSHAAand40BitRC2-CBC.

/OUTPUT_FILE=*prefix*	Use *prefix* as the prefix for all output filenames. Private key filenames will be *prefix*.SSH2 and PKCS#10 files will be *prefix*.PKCS10.
/PRIVATE_KEY=*keyname*	Use *keyname* as the private key.
/SUBJECT=*"subject"*	(PKCS#10 only) Use *subject* as the certificate subject.
/VERSION	Display the version of CERTTOOL.

Example:

```
$ CERTTOOL /PK10 /SUBJECT=("cn=john doe,cn=lima,cn=beans"-
$_ /PRIVATE_KEY=DKA0:[JOHENDOE.SSH2]ID_DSA_1024_A
PKCS#10 creation successful.
Wrote certificate request to output.pkcs10.
```

CERTVIEW

CERTVIEW can be used to view certificates and check their validity. This tool can also be used to output the data in format that is suitable for insertion in the SSH2_DIR:SSHD2_CONFIG configuration file.

Format

certview [*options*] *certificate* [, *certificate*, ..., *certificate*]

Valid Options

/COMMENT	Prepend information lines with # (comment mark)
/DEBUG=*n*	Set debug level to *n*
/FORMAT_OUTPUT	Output data in a format suitable for insertion to user-map
/HELP	Display help
/QUIET	Don't display certificate information
/VALIDATE=*certificate*	Validate using the CA certificate *certificate*
/VERBOSE	Increase verbosity (display extensions).
/VERSION	Display version information

Example:

```
$ CERTVIEW MYCERT_PKCS7.P7B-1_SSH2_CRT
Certificate MYCERT_PKCS7.P7B-1_SSH2_CRT
Certificate issuer ........... : MAILTO=foo@bar.com, C=US, ST=MA,
L=Framingham, CN=FOOCA
Certificate serial number .... : 206680290271582356976177697926629044 21
Certificate subject .......... : MAILTO=foo@bar,com, C=US, ST=MA,
L=Framingham, CN=FOOCA
```

CMPCLIENT

Allows users to enroll certificates. It will connect to a CA (certification authority) and use the CMPv2 protocol for enrolling a certificate. The user may supply an existing private key when creating the certification request or allow a new key to be generated.

Format

`cmpclient [`*`options`*`]/ca_access_url="`*`url`*`"/subject="`*`subject`*`" cert-file [`*`private-key`*`]`

Command Parameters

`url`	Specifies the URL for the Certification Authority
`Subject`	Specifies the subject name for the certificate. For example, `c-ca,o=acme,ou=development,cn=Bob Jones`
`Cert file`	Specifies the file the certification is written to.
`Private key`	Specifies the private key to be written to.

Valid Options

`/BASE=`*`name`*	Specify base prefix for the generated files.
`/BITS=`*`n`*	Specify the key length in bits.
`/CA__URL="`*`url`*`"`	Specify the URL of the Certification Authority.
`/DEBUG=`*`n`*	Set debug to level *n* (0-60).
`/ENROLLMENT_PROTOCOL=`*`prot`*	Use specified enrollment protocol (SCEP or CMP).
`/EXTENSIONS`	Enable extensions in the subject name.

/GENERATE_KEY	Generate a new private key.
/HELP	Print this help text.
/PROXY_URL"=*url*"	Specify the URL of the HTTP proxy server URL to be used when connecting to the certification authority.
/REFNUM=*refnum:key*	Specify the CMP enrollment reference number and key.
/SOCKS_SERVER="*url*"	Specify the URL of the SOCKS server to be used when connecting to the certification authority.
/SUBJECT="*subject*"	Specifies the subject name for the certificate.
/TYPE=rsa|dsa	Specify the key type to generate (default: rsa)
/USAGE_BITS=*n*	Specify the key usage bits.
/VERSION	Print the version information for this program.

Examples:

1. Enroll a certificate and generate a DSA private key:

```
$ cmpclient/type=dsa/generate_key/base=mykey/refnum=1234:abc -
_$ /ca_access_url="http://www.ca-auth.domain:8080/pkix/"-
_$ /subject="c=us,o=foobar,cn=Dilbert Dogbert" ca-certification.crt
```

This will generate a private key called mykey.prv and a certificate called mykey-0.crt.

2. Enroll a certificate using a supplied private key and provide an e-mail extension:

```
$ cmpclient/base=mykey/refnum=12345:abcd -
_$ /ca_access_url="http://www.ca-auth.domain:8080/pkix/"-
_$ /subject="c=us,o=foobar,cn=Dilbert Dogbert:email=foo@bar.com" -
_$ ca-certification.crt my_private_key.prv
```

This will generate and enroll a certificate called mykey-0.crt.

Note: SSH stores and uses software certificates in DER encoded binary format. You can use `sshkeygen` to import and convert PKCS#12 packages (`/pkcs_convert=`*`file`*) into a private key/certificate pair, X.509 format private key into an SSH private key (`/x509_convert=`*`file`*), or PKC#7 into a certificate (`/extract_certs=`*`file`*).

Public Key Subsystem

The public key subsystem and assistant that can be used to add, remove and list public keys stored on a remote server. The public key assistant and server are based upon a recent IETF draft, so other implementations of SSH may not yet offer this functionality.

The public key assistant can be started with:

```
$ PUBLICKEY_ASSISTANT [qualifiers] [[user@]host[#port]]
```

Public Key Assistant Commands

ADD *key_file_name*
Transfers the key file name to the remote system. The file name specified is expected to be in the SSH2_CONFIG directory from the user's login directory. e.g., ADD ID_DSA_1024_A.PUB will transfer the public key in ID_DSA_1024_A.PUB to the remote system and updates the AUTHORIZATION. file on the remote system to include this key name.

CLOSE
Closes the connection to the remote system

DEBUG {no | *debug_level*}
Sets debug level (like in SFTP2).

DELETE *key finger*-print
Deletes the key that matches the fingerprint specified. It is necessary to do a LIST command before this to get a list of the fingerprints (and for the program to build its internal database mapping fingerprints to keys).

EXIT
Exits the program.

HELP
Displays a summary of the commands available.

LIST
Displays the fingerprint and attributes of keys stored on the remote system. The attributes that are listed will vary with key.

OPEN [*user*@]*host*[#*port*]
Opens a connection to a remote public key subsystem.

QUIT
Quits the program.

UPLOAD *key_file_name*
Transfers the key file name to the remote system. The file name specified is expected to be in the SSH2_CONFIG directory from the user's login directory.

VERSION [*protocol_version*]
Displays or sets the protocol version to use. The protocol version can only be set before the OPEN command is used. The default version is 1.

Qualifiers

/BATCHFILE
Provides file with public key assistant commands to be executed. Starts SSH2 in batch mode. Authentication must not require user interaction.

/CIPHER
Selects encryption algorithm(s).

/COMPRESS
Enables SSH data compression.

/DEBUG
Sets debug level (0-99).

/HELP
Displays a summary of the qualifiers available.

/MAC=(*mac-1*,…,*mac-n*)
Selects MAC algorithm(s).

/PORT
Tells SFTP2 which port SSHD2 listens to on the remote machine.

/VERBOSE
Enables verbose mode debugging messages. Equal to `/debug=2`. You can disable verbose mode by using `debug disable`.

/VERSION
Displays version number only.

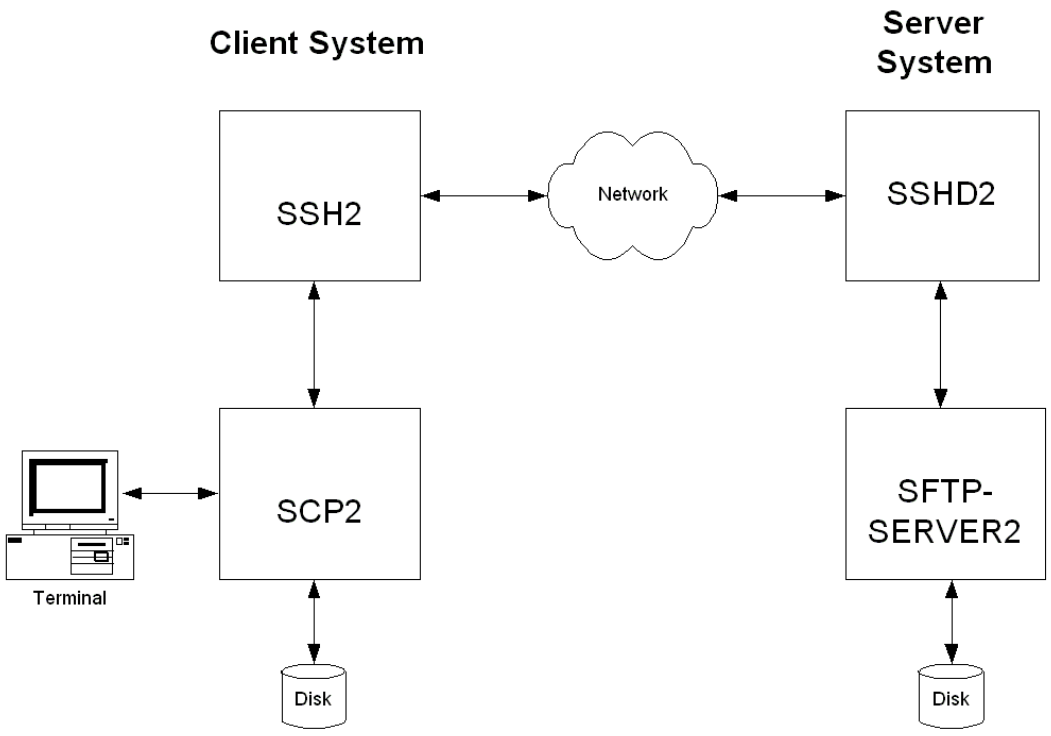
Other Implementations

VanDyke includes this in their SecureFX and VShell products. VanDyke also has a patch available for a server for OpenSSH.

16. Secure File Transfer

There are three methods to do secure file transfer: SCP2, SFTP2, and FTP over SSH2. SCP2 and SFTP2 communicate with SSH2 for authentication and data transport (which includes encryption) to remote systems and to activate the `SFTP-SERVER2` image. An SCP1 server is provided for compatibility with OpenSSH SCP.

The following diagram illustrates the relationship among the client and server portions of an SCP2 or SFTP2 file transfer:



SCP file transfers are different from FTP file transfers. With FTP a file can be transferred as ASCII, BINARY, RECORD, or in OpenVMS format (if MultiNet or TCPware is in use). In SCP the primary transfer format is BINARY. Also, the defined syntax for a file specification is UNIX syntax. Due to these restrictions, files that are transferred from dissimilar systems may or may not be useful. ASCII transfers are done by searching the transferred data for the specified newline sequence and making the specified substitution. Process Software has used methods available in the protocol to attempt to improve the chances that files will be useful upon transfer. The SSH File Transfer Protocol is an evolving specification, and some implementations may not support all options available in the protocol, or worse, not tolerate some optional parts of later versions of the protocol.

Process Software has used the defined extensions in the protocol to transfer information about the VMS file header characteristics such that when a file is transferred between two VMS systems running MultiNet v4.4 or higher, TCPware v5.6 or higher, and/or SSH for OpenVMS, the file header information will also be transferred and the file will have the same format on the destination system as it had on the source system. Also, when a text file is transferred to a non-VMS system, a method has been provided to convert those files that can be translated into a format that will be usable on the remote system. Files that are converted from non-VMS systems are stored as stream files on the VMS system, which provides compatibility for text files from those systems. Filenames are SRI-encoded when files are stored on ODS-2 disks.

SCP-SERVER1

The `SCP-SERVER1` program is used when a system with OpenSSH initiates an SCP command. OpenSSH uses RCP over SSH2 instead of the SFTP protocol. `SCP-SERVER1` will always convert VMS text files (if possible) when copying a file from VMS. Converted VMS text files may have some trailing nulls at the end of them, due to the RCP protocol not being able to tolerate a file that comes up short of the reported size. `SCP-SERVER1` (and `SFTP-SERVER2`) use sophisticated methods to estimate the amount of user data in the file to minimize this. On ODS-5 disks the estimation routine uses the file size hint if it is valid. On ODS-2 disks (and ODS-5 without a valid size hint), the size of the file and file characteristics are used to estimate the amount of user data. The method provides as accurate an estimate as possible without actually reading the file and never underestimates the amount of data in the file. Underestimating would cause significant problems as the programs use the size of the file to determine how much data to expect.

SCP2

Usage

`SCP2 [qualifiers] [[user@]host[#port]::]file [[user@]host[#port]::]file`

> **Note:** The source and destination file specification must be quoted if they contain a user specification or a non-VMS file specification.

Qualifiers

Qualifier	Description
`/ASCII[=newline convention]`	Newline convention is one of dos, mac, unix, vms, or sftp. The newline convention specified is the newline convention to use if a newline convention is not specified by the server. Allowed values: `dos` (\r\n), `mac` (\r) , `unix` (\n), `vms` (\n) , `sftp` (\r\n). Default = `unix`.
`/BATCH`	Starts SSH2 in batch mode. Authentication must be possible without user interaction.
`/BUFFER_SIZE=integer`	Number of bytes of data to transfer in a buffer. Default is 7500. Minimum value is 512.
`/CIPHER=(cipher-1,…,cipher-n)`	Selects an encryption algorithm(s).
`/COMPRESS`	Enables SSH data compression.
`/CONCURRENT_REQUEST=n`	Number of concurrent read requests to post to the source file. Default is 4.
`/DEBUG=level`	Sets a debug level. (0-99)

/DIRECTORY	Forces the target to be a directory.
/HELP	Displays the help text.
/IDENTITY_FILE=*file*	Identifies the file for public key authentication.
/OVERWRITE	Overwrite existing file instead of deleting first.
/PORT=*number*	Tells SCP2 which port SSHD2 listens to on the remote machine.
/PRESERVE	Preserves file attributes and timestamps.
/NOPROGRESS	Does not show progress indicator.
/QUIET	Does not display any warning messages.
/RECORD	Open the source file in VMS Record mode if possible. This is equivalent to record mode transfer in SFTP2. The file is transferred as a stream of records with no carriage control added between them.
/RECURSIVE	Processes the entire directory tree.
/REMOVE	Removes the source files after copying.
/TRANSLATE_VMS= *(ALL, NONE, VARIABLE, FIXED, VFC)*	Selects the VMS text files to be translated (default=ALL). Note that /ASCII performs a similar function and may be supported in other SCP products.
/VERBOSE	Displays verbose debugging messages. Equal to /debug=2.
/VERSION	Displays the version number only.
/VMS	Negotiates the ability to transfer VMS file information.

> **Note:** /ASCII, /VMS and /TRANSLATE_VMS are mutually exclusive

File Specifications

The source and destination strings are changed to lowercase unless they are enclosed in quotes, in which case they are left the same. File specification must be in UNIX format for remote systems, unless the remote system is running TCPware 5.6 or higher, MultiNet v4.4 or higher, or SSH for OpenVMS, and /VMS or /TRANSLATE_VMS (source files only) are used. UNIX format file specifications need to be enclosed in quotes (") if they contain the / character to prevent the DCL parsing routines from interpreting the string as a qualifier.

Qualifiers

/ASCII[=*newline convention*]
Uses the newline convention specified if the server does not specify a newline convention. Available conventions are: dos (\r\n), mac (\r) , unix (\n), vms (\n) , sftp (\r\n). Default = unix.

/BATCH
Starts SSH2 in BATCH mode. When SSH2 is running in BATCH mode it does not prompt for a password, so user authentication must be performed without user interaction.

/BUFFER_SIZE=*integer*
Number of bytes of data to transfer in a buffer. Default is 7500.

/CIPHER=(*cipher,...,cipher-n*)
Lets you select which SSH2 cipher to use.

/COMPRESS
Enables SSH2 data compression. This can be beneficial for large file transfers over slow links. The compression level is set by the client configuration file for SSH2.

/CONCURRENT_REQUEST=*integer*
Number of concurrent read requests to post to the source file. Default is 4.

/DEBUG
Enables debugging messages for SCP2 and SSH2. Higher numbers get more messages. The legal values are between 0 (none) and 99. Debugging for SFTP-SERVER2 is enabled via the TCPWARE_SSH_SFTP_SERVER_DEBUG logical.

/DIRECTORY
Informs SCP2 that the target specification should be a directory that the source file(s) will be put in. This qualifier is necessary when using wildcards in the source file specification, or /RECURSIVE.

/HELP
Displays command qualifier list and parameter format.

/IDENTITY_FILE=*file*
Specifies the identity file that SSH2 should use for public key authentication.

/PORT=*number*
Specifies the port that SSH2 uses on the remote system. Note that if both the source and destination files are remote, this value is applied to both. If SSH2 is available on different ports on the two systems, then the #port method must be used.

/PRESERVE
Sets the Protection, Owner (UIC), and Modification dates on the target file to match that of the source file. The adjustment of timestamps for time zones is dependent upon the logical SYS$LOCALTIME being set correctly. This is defined automatically on VMS V7 and can be defined similarly on earlier versions of VMS. /PRESERVE is not very useful when the target machine is a VMS system as VMS does not provide runtime library calls for setting the file attributes (owner, protection) and timestamps. Note that the VMS modification date (not the creation date) is propagated to the remote system. When files are copied between two VMS systems and /VMS is used /PRESERVE is implied and the process of transferring VMS attributes preserves the information about the protection, dates, and file characteristics.

/NOPROGRESS
SCP2, by default, updates a progress line at regular intervals when it is run interactively to show how much of the file has been transferred. This qualifier disables the progress line.

/QUIET

Disables warning messages. Note that it does not disable warning messages from SFTP-SERVER2, which return on the error channel.

/RECORD
Open the source file in VMS record mode. This copies the source file to the destination as records converted to a stream of bytes without any carriage control between records. This is equivalent to RECORD mode transfer in SFTP.

/RECURSIVE
Copies all of the files in the specified directory tree. Note that the top level directory on the local system is not created on the remote system. Only the most recent version is copied unless in VMS mode and the TCPWARE_SFTP_VMS_ALL_VERSIONS logical is defined to be TRUE.

/REMOVE
Deletes the source files after they have been copied to the remote system.

/TRANSLATE_VMS
Translates VMS text files in the copying process to byte streams separated by linefeeds because the defined data transfer format for SCP2 is a binary stream of bytes.

/TRANSLATE_VMS is only applicable to the source specification. If a remote source file is specified, then that system must be running MultiNet v4.4 or higher, TCPware 5.6 or higher, or SSH for OpenVMS. If /TRANSLATE_VMS is specified with no value, then VARIABLE, FIXED, and VFC (Variable, Fixed Control) files are translated to stream linefeed files. If the value is NONE, no files are translated. VARIABLE, FIXED, and VFC can be combined in any manner. The SFTP-SERVER2 process uses the value of the logical TCPWARE_SFTP_TRANSLATE_VMS_FILE_TYPES to determine which files should be translated automatically. This is a bit mask with bit 0 (1) = FIXED, bit 1 (2) = VARIABLE, and bit 2 (4) = VFC. These values can be combined into a number between 0 and 7 to control which files are translated.

> **Note:** Due to the structure of the programs, the SCP2 program uses this logical if the /TRANSLATE_VMS qualifier has not been specified.

/VERBOSE
Displays debugging messages that allow the user to see what command was used to start up SSH and other basic debugging information. Note that debugging information can interfere with the normal display of the progress line. Equivalent to /DEBUG=2.

/VERSION
Displays the version of the base SCP2 code.

/VMS
Transfers VMS file information similar to that transferred in OVMS mode in FTP such that VMS file structure can be preserved. All of the information transferred in FTP OVMS mode is transferred along with the file creation date and protection. Timestamps are not adjusted for time zone differences in VMS transfers. If the file is a contiguous file, and it is not possible to create the file contiguously, and the logical TCPWARE_SFTP_FALLBACK_TO_CBT has the value of TRUE, YES, or 1, SFTP-SERVER2 attempts to create the file Contiguous, Best Try.

The logical name TCPWARE_SCP2_VMS_MODE_BY_DEFAULT can be defined to specify that /VMS should be the default unless /NOVMS or /TRANSLATE_VMS are specified. /VMS and /TRANSLATE_VMS cannot be used on the same command line. If /VMS is not specified, but the logical is set to enable it by default, a /TRANSLATE_VMS on the command line will take precedence.

Note that even though SCP2 & SFTP-SERVER2 pass the request for VMS file transfers or to translate a VMS file in a manner that is consistent with the protocol specification, other implementations may not handle this information well. Since there is no error response present at that point in the protocol, the program hangs. To prevent it from hanging forever, the logical TCPWARE_SCP2_CONNECT_TIMEOUT is checked to see how long SCP2 should wait for a response when establishing the connection. The format for this logical is a VMS delta time. The default value is 2 minutes. If SCP2 times out before a connection is established with SFTP-SERVER2 and /VMS or /TRANSLATE_VMS were specified, a warning message is displayed, and the initialization is tried again without the request for VMS information (or /TRANSLATE_VMS). This retry is also subject to the timeout, and if the timeout happens again, then SCP2 exits. This helps for implementations that ignore the initialization message when information they do not recognize is present; implementations that abort will cause SCP2 to exit immediately.

Logicals

For the following logicals, all that start TCPWARE_SFTP apply to the SCP2 client, SFTP2 client and SFTP2 server.

TCPWARE_SFTP_FALLBACK_TO_CBT
When defined and a VMS file transfer is being performed, this logical creates a Contiguous file if that file has Contiguous characteristics. The file will be created as Contiguous Best Try if there is insufficient space to create it as Contiguous.

TCPWARE_SFTP_TRANSLATE_VMS_FILE_TYPES
This is a bit mask that determines which VMS file types should be translated when not operating in VMS mode.

- Bit 0 (1) = FIXED
- Bit 1 (2) = VARIABLE
- Bit 2 (4) = VFC

The values are:

- 0 (zero) = NONE
- 7 = ALL

Note that this logical affects SCP2 as well as the server, as SCP2 has the server built into it for handling local file access. If this logical is not defined, the value 7 will be used.

TCPWARE_SCP2_CONNECT_TIMEOUT
This logical defines a number specifying how long SCP2 should wait for a response to the INITIALIZE command from the server program. This is a VMS delta time number. The default is 2 minutes.

TCPWARE_SCP2_VMS_MODE_BY_DEFAULT
When defined, this logical chooses the `/VMS` qualifier if `/TRANSLATE_VMS` or `/NOVMS` has not been specified.

TCPWARE_SFTP_RETURN_ALQ
When defined and files are being transferred in VMS mode, this logical includes the Allocation Quantity for the file in the file header information. This is disabled by default because copying a small file from a disk with a large cluster size to a disk with a small cluster size causes the file to be allocated with more space than necessary. You have the option of retaining the allocated size of a file if it was allocated the space for a reason. Some combinations of file characteristics require that the Allocation Quantity be included in the file attributes; this is handled by SCP2/SFTP-SERVER2.

TCPWARE_SSH_SCP_SERVER_DEBUG

Enables debugging messages for the SCP-SERVER1 image that provides service to SCP commands that use the RCP over SSH2 protocol (OpenSSH). When this is defined, the file SCP-SERVER.LOG is created in the user's login directory. These files are not purged. Larger values yield more debugging information.

TCPWARE_SSH_SFTP_SERVER_DEBUG
Enables debugging messages for the SFTP-SERVER2 image that provides service to SCP2 commands that use the SFTP protocol. When this is defined, the file SFTP-SERVER.LOG is created in the user's login directory. These files are not purged. Larger values yield more debugging information

TCPWARE_SFTP_MAXIMUM_PROTOCOL_VERSION
This logical can be used to limit the version of the SSH protocol that the SFTP client and server use. This can sometimes provide a work-around for problems encountered with different implementations of the protocol. The default value is 4. Protocol versions 2 and 3 are also used by popular implementations.

TCPWARE_SFTP_VMS_ALL_VERSIONS
This logical controls whether or not all versions of a file are returned. Defining this logical will cause all versions to be returned, the undefined behavior is to only return the name of the file without a version. The default is to return only one filename without the version number.

TCPWARE_SFTP_NEWLINE_STYLE
This logical controls the newline style that SFTP uses. Which can be helpful in transferring text files. The values are: UNIX <lf>, VMS <lf>, MAC <cr>. If the logical is not defined, or defined to any other value, then <cr><lf> will be used for the text line separator as documented in the SSH File Transfer specification.

TCPWARE_SFTP_CASE_INSENSITIVE
This logical causes SFTP to treat filenames in a case insensitive manner when it is defined.

TCPWARE_SFTP_ODS2_SRI_ENCODING
This logical controls whether or not SRI encoding is used for filenames on VMS ODS-2 disks. If the logical is not defined, or is defined to TRUE, YES, or 1 (the number one) then SRI encoding is used on ODS-2 disks for filenames that contain uppercase letters and special characters.

TCPWARE_SFTP_FILE_ESTIMATE_THRESHOLD
This logical controls the minimum number of blocks that a text file must be for an estimated transfer size to be returned instead of an exact size. The default is to estimate the transfer size for all text files.

TCPWARE_SFTP_DEFAULT_FILE_TYPE_REGULAR
If this logical is defined to TRUE, YES or 1 (the number one), then the SFTP server will use a default file type of REGULAR instead of UNKNOWN for OPEN operations. This can correct problems with filenames without a . (dot) in them getting .dir added to them. The filename will appear with a . at the end of the name in directory listings.

TCPWARE_SFTP_*username*_CONTROL
This logical can be defined /SYSTEM to any combination of NOLIST, NOREAD, NOWRITE, NODELETE, NORENAME, NOMKDIR, NORMDIR, to restrict operations for the username in the logical. NOWRITE will disable PUT, DELETE, RENAME, MKDIR, RMDIR; NOREAD will disable GET and LIST.

TCPWARE_SFTP_*username*_ROOT
The logical TCPWARE_SFTP_*username*_ROOT can be defined /SYSTEM to restrict the user to the directory path specified. Subdirectories below the specified directory are allowed.

SSH_SFTP_LOG_SEVERITY
The logical SSH_SFTP_LOG_SEVERITY can be defined /SYSTEM to 20000 to log file transfers or 30000 to log all SFTP operations.

SSH2_SFTP_LOG_FACILITY
The logical SSH2_SFTP_LOG_FACILITY must also be defined /SYSTEM to specify the logging class that is used with OPCOM.

Values below 5 will use the network class; 5 will use OPER1, 6 will user OPER2, etc. The maximum value that can be specified is 12, which will use OPER8.

TCPWARE_SFTP_SEND_VENDOR_ID
If this logical is defined to “No”, “False” or “0” (zero), then the SFTP2 client will not send the extended command containing the vendor-id upon completion of version negotiation with the server.

SFTP2

File Specifications

File specification must be in UNIX format for remote systems, unless /VMS transfers are being used.

Usage

`SFTP2 [qualifiers] [[user@]host[#port]]`

If the `user@` is included in the remote system specification, the specification must be enclosed in quotes.

Qualifiers

Qualifier	Description
`/BATCHFILE=<file specification>`	Provides file with SFTP commands to be executed. Starts SSH2 in batch mode. Authentication must not require user interaction.
`/BUFFER_SIZE=integer`	Number of bytes of data to transfer in a buffer. Default is 7500.
`/CIPHER=(cipher-1,...,cipher-n)`	Selects encryption algorithm(s).
`/COMPRESS`	Enables SSH data compression.
`/CONCURRENT_REQUEST=integer`	Number of concurrent read requests to post to the source file. Default is 4.
`/DEBUG=level`	Sets debug level (0-99).
`/HELP`	Displays help.

/MAC=*(mac-1,...,mac-n)*	Select MAC algorithm(s).
/NOPROGRESS	Do not show progress indicator.
/PORT	Tells SFTP2 which port the SSHD2 server is listening on.
/VERBOSE	Enables verbose mode debugging messages. Equal to /debug=2. You can disable verbose mode by using debug disable.
/VERSION	Displays version number only.
/[NO]VMS	Negotiates ability to transfer VMS file information. VMS transfer mode will be automatically negotiated if SFTP2 detects that the server is capable of doing VMS transfers unless /NOVMS is specified.

SFTP2 Commands

SFTP2 Command	**Description**
ASCII[{-s |*remote* [*local*]}]	With -s option, shows current newline convention. *remote nl conv* sets remote newline convention. *local nl conv* operates on local side, but is not as useful (the correct local newline convention is usually compiled in, so this is mainly for testing). You can set either of these to ask, which will cause sftp to prompt you for the newline convention when needed. With the exception of the -s option, this command sets transfer mode to ascii. Available conventions are dos, unix, sftp, vms, or mac, using "\r\n", "\n", "\r\n", "\n" and "\r" as newlines, respectively.

	Note that some implementations of SFTP may check to see if a file can be transferred in ASCII mode before doing so, and return errors for files that cannot be transferred. SSH for OpenVMS, MultiNet, and TCPware make this check.
AUTO	Sets the transfer mode (ASCII or BINARY) to depend upon the extension of the file specification.
BINARY	Sets the transfer mode to be binary. (This is the default.)
BUFFERSIZE *[number]*	Sets the size of the buffer used for file transfer. A larger buffer size helps speed large transfers. Displays the current buffer size when no parameter is specified.
CD *directory*	Changes current directory on remote system. VMS file specifications may be used when operating in VMS mode. A logical name must include the trailing colon so that it can be recognized as such. SFTP from other vendors cannot use VMS specifications due to the way that SFTP works.
CHMOD [-R] *mode file* [*file…*]	Change the protection on a file or directory to the specified octal mode. (UNIX values). -R recurses over directories.
CLOSE	Closes connection to the remote server.
DEBUG {disable | no |*debug_level}*	Sets the debug level for SFTP2. It does not change the current debug level for SSH2 for an existing connection but will be used with SSH2 for a new connection. With disable or no, this disables all debugging current sessions for SFTP2.

DELETE *file*	Removes the specified file from the remote system.
DIRECTORY [*file* | *directory]*	Displays the contents of the current directory or specified directory in VMS format when the transfer mode is VMS. File names are displayed as they would be with a DIRECTORY command from DCL.
EXIT	Exits SFTP client.
GET [-p] *file*1 [*file2* …]	Retrieves the specified file(s) from the remote system and stores it in the current working directory on the local system. File names are case sensitive and in UNIX format. When operating in VMS mode, either UNIX or VMS-style file specifications can be used. Directories are recursively copied with their contents. Multiple files may be specified by separating the names with spaces. If -p is specified, then SFTP attempts to preserve timestamps and access permissions. Note that a target filename cannot be provided.
GETEXT	Displays the list of file extensions to use ASCII transfers when in AUTO mode. The initial value is txt,htm*,pl,php*
HELP	Displays help on commands.
LCD *directory*	Changes the current directory on the local system. VMS file specifications may be used when in VMS mode.
LCHMOD [-R] *mode file* [*file* …]	Change the protection on a file or directory on the local connection to the specified octal mode. (Unix values). -R recurses over directories.
LCLOSE	Close the local connection.

LDELETE *file*	Removes the specified file from the local system. VMS file specifications may be used when in VMS mode.
LDIRECTORY [*file* | *directory*]	Displays the contents of the current directory for the local system in VMS format when the transfer mode is VMS. File names are displayed as they would be with a DIRECTORY command from DCL.
LLS [*file* | *directory*]	Displays the contents of the current directory or specified directory in UNIX format. Lists the names of files on the local server. For directories, contents are listed. See LS for options and more details.
LLSROOTS	Like LSROOTS, but for the local side.
LMKDIR *directory*	Creates the specified directory on the local system.
LOCALOPEN {[*user@host*[*#port*] | -1}	Tries to connect the local side to the host *host*. If successful, lls and friends will show the contents of the filesystem on that host. With the -l option, connects to the local filesystem (which doesn't require a server). There is an implied LOCALOPEN -l when SFTP2 starts up. Note that an implicit LOCALOPEN is done when SFTP2 starts, so the only time that a user needs to do a LOCALOPEN is when neither directory tree is immediately accessible. OPEN is the command that is generally used to establish the connection with the remote system. LOPEN is a synonym for LOCALOPEN.
LPWD	Displays the current working directory on the local system.

LREADLINK *path*	Provided that *path* is a symbolic link, shows where the link is pointing to. This command is not supported for VMS.
LRENAME *oldfile newfile*	Renames a file on the local system.
LRM *file*	Removes the specified file from the local system. VMS file specifications may be used when in VMS mode.
LRMDIR *directory*	Deletes a directory on the local system.
LS [-R] [-l] [-S] [-r] [*file …*]	Displays the contents of the current directory or specified directory in UNIX format. Lists the names of files on the remote server. For directories, contents are listed. When the -R is given, directory trees are listed recursively. (By default, subdirectories of the arguments are not visited.) When the -l option is given, permissions, owners, sizes, and modification times are also shown. When the -S option is specified, sorting is based upon file size instead of alphabetically. The -r option reverses the sort order. When no arguments are given, it assumes that the contents of "." (current working directory) are being listed. Currently, the options -R and -l are mutually incompatible. LS will fill a screen with output, then wait for the user to decide if they want more or have seen enough.
LSROOTS	Displays the virtual roots of the server. (This VanDyke Software's V Shell extension. Without this you can't know the filesystem structure of a V Shell server). This is also a VMS extension to display the roots (devices) on the VMS system. Though the commands are the same, the information provided is not compatible with what is displayed by VanDyke Software's Secure FX.
LSYMLINK *targetpath linkpath*	Like SYMLINK, but for the "local" side.

`MGET [-p] file1 [file2 …]`	Retrieves multiple files from the remote system and stores them in the current working directory on the local system. If `-p` is specified, then SFTP attempts to preserve timestamps and access permissions.
`MKDIR directory`	Creates the specified directory on the remote system.
`MPUT [-p] file1 [file2 …]`	Stores multiple files in the current working directory on the remote system. File names are case-sensitive and in UNIX format. When operating in VMS mode, either UNIX or VMS-style file specifications can be used. Directories are recursively copied with their contents. Multiple files may be specified by separating the names with spaces. If -p is specified, then SFTP attempts to preserve timestamps and access permissions.
`OPEN {-1| [user@]host[#port]}`	Tries to connect to the host *`host`*. Or with the `-1` option, connects the remote side to the local filesystem (which doesn't require a server).
`PUT [-p] file1 [file2 …]`	Stores the specified file in the current working directory on the remote system. File names are case-sensitive and in UNIX format. When operating in VMS mode, either UNIX or VMS-style file specifications can be used. Directories are recursively copied with their contents. Multiple files may be specified by separating the names with spaces. If -p is specified, then SFTP attempts to preserve timestamps and access permissions.

	Note that a target filename cannot be provided.
PWD	Displays the current working directory on the remote system. Displayed in VMS format when in VMS mode; otherwise displayed in UNIX format.
QUIT	Exits SFTP client.
READLINK <*targetpath*><*linkpath*>	Provided that *path* is a symbolic link, shows where the link is pointing to. Not valid for VMS systems as VMS does not have symbolic links.
RECORD	Enters record transfer mode if the server supports Process Software's record open. The direction in which record transfer mode is possible will be displayed in response to this command. In record transfer mode the source file is opened as binary records and the destination file is opened as binary. This produces the same effect as TCPware's FTP server BINARY transfer when a BLOCK_SIZE has not been specified, and can be used to transfer a file that contains VMS records to a system that can only handle "flat" files.
RENAME *oldfile newfile*	Renames file on the remote system.
RM *file*	Removes the specified file from the remote system.
RMDIR *directory*	Deletes a directory on the remote system.
SETEXT *ext1* [*ext2* …]	Sets the list of file extensions to use ASCII transfers when in AUTO mode. Individual file extensions must be separated by spaces.
STATUS	Shows the transfer mode, remote server name, and remote server version. The current newline sequence is displayed if operating in ASCII or AUTO mode.

SYMLINK *targetpath linkpath*	Creates symbolic link *linkpath*, which will point to *targetpath*. Not valid for VMS systems as VMS does not have symbolic links.
VERBOSE	Enables verbose mode (identical to /DEBUG=2 command-line option). You may later disable verbose mode by debug disable.
VMS	Sets the transfer mode to include VMS file information.

Logicals

The following logicals are specific to SFTP2.

TCPWARE_SFTP_VMS_MODE_BY_DEFAULT
When defined to TRUE, YES, or 1, this logical chooses the /VMS qualifier if /NOVMS has not been specified.

Configuration File Parameters

The system wide configuration file (SSH2_DIR:SSH2_CONFIG.) or the user's configuration file (SYS$LOGIN:[.SSH2]SSH2_CONFIG.) can be used to specify the following parameters. The user's configuration file takes precedence over the system configuration file.

FilecopyMaxBuffers	This is equivalent to the /CONCURRENT_REQUEST qualifier on the SFTP2 or SCP2 command line. The command line qualifier will supersede any value in the configuration file.
FilecopyMaxBuffersize	This is equivalent to the SFTP2 BUFFERSIZE command or the SCP2 /BUFFER_SIZE qualifier. The command or qualifier takes precedence.

The system server configuration file (SSH2_DIR:SSHD2_CONFIG.) can include parameters to control which users can perform remove SSH commands (including SSH terminal sessions) as well as SFTP2 access.

Terminal.AllowUsers	Allow users in the specified list to create SSH2 terminals and do interactive commands
Terminal.DenyUsers	Prevent users in the specified list from creating SSH2 terminals and performing interactive commands. The users can still use the SFTP2, SCP1 and Public Key servers.
Terminal.AllowGroups	Allow groups in the specified list to create SSH2 terminals and do interactive commands
Terminal.DenyGroups	Prevent groups in the specified list from creating SSH2 terminals and performing interactive commands. The groups can still use the SFTP2, SCP1 and Public Key servers.

FTP over SSH

SSH2 can be used to set up port forwarding that can be used for FTP. This allows users to use the richness of the FTP command set to access files on a remote system and have their control and data information encrypted. The command format to set up the SSH port forwarding is:

```
$ ssh remote_host_name/local_forward=
  (" " "ftp/forwarded_port_number:localhost:21" " ")
```

The usual SSH authentication mechanisms come into play, so there may be a request for a password and a terminal session is established to the remote host. As long as this terminal session is alive, other users on the local system can use FTP to access the remote system over an encrypted channel. The location of the quotes is important, as it is necessary to prevent DCL from interpreting the / in the local forwarding information as the start of a new qualifier, and SSH2 does not know or expect to find the ( ) around the forwarding information. Note that the "localhost" inside of the forwarding string is important, as it will make the connection to FTP on the remote system come from localhost, which will then allow FTP to open the data port.

When a user desires to use an encrypted FTP connection, the following sequence of commands would be issued:

```
PORT forward_port_number
OPEN LOCALHOST
```

Normal FTP authentication takes place and multiple FTP sessions may use a single forwarded port. The FTP protocol filter in SSH2 scans the FTP command stream for the FTP `PORT` and `PASV` commands and their replies, and makes substitutions in these commands and replies to use a secure data stream through the SSH2 session that has been set up. This command will establish an encrypted FTP session with the remote host that the SSH connection is sent to.

To allow a single system to act as a gateway between two networks, add `/ALLOW_REMOTE_CONNECT` to the SSH command that initiates the connection.

Appendix A. References

Introduction

This appendix lists documentation to which you can refer for additional details about TCPware for OpenVMS, TCP/IP protocol suite, networking concepts, and related subjects.

TCPware Documentation

Be sure you have the following additional TCPware for OpenVMS documents available for reference:

- *Installation & Configuration Guide*
- *Management Guide*
- *Network Control Utility (NETCU) Command Reference*
- *Programmer's Guide*

Requests for Comments (RFCs)

Requests for Comments (RFCs) documents contain the specifications for all internet protocols. Unless specifically noted otherwise on the RFC itself, all RFCs are for unlimited distribution.

You can obtain RFCs by going to the http://www.rfcs.org web site.

The below table lists the RFCs containing the protocol specifications implemented by TCPware.

Title	**RFC #**
User Datagram Protocol (STD 6)	768
Internet Protocol: DARPA Internet Program Protocol Specification	791
Internet Control Message Protocol (see also RFC 950)	792
Transmission Control Protocol	793
Simple Mail Transfer Protocol (STD 10)	821

Standard for the Format of Text Messages (STD 11)	822
An Ethernet Address Resolution Protocol	826
TELNET Protocol Specification (STD 8)	854
TELNET Option Specification (STD 8)	855
TELNET Binary Transmission (STD 27)	856
TELNET Echo Option (STD 28)	857
TELNET Suppress Go Ahead Option (STD 29)	858
Echo Protocol (STD 20)	862
Discard Protocol (STD 21)	863
Character Generator Protocol (STD 22)	864
Quote of the Day Protocol	865
Daytime Protocol (STD 25)	867
Time Protocol (STD 26)	868
TELNET End of Record Option	885
Trailer Encapsulations	893
A Standard for the Transmission of IP Datagrams over Ethernet Networks	894
Reverse Address Resolution Protocol	903
Broadcasting Internet Datagrams (STD 5)	919
Broadcasting Internet Datagrams in the Presence of Subnets (STD 5)	922

Internet Standard Subnetting Procedures (STD 5)	950
Bootstrap Protocol (BOOTP)	951
File Transfer Protocol (STD 9)	959
Mail Routing and the Domain System (STD 14)	974
XDR: External Data Representation Standard	1014
Domain Administrators Guide	1032
Domain Administrators Operations Guide	1033
Domain Names: Concepts and Facilities	1034
A Standard for the Transmission of IP Datagrams over IEEE 802 Networks	1042
Internet Protocol on Network Systems HYPERchannel Protocol Specification	1044
A Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP	1055
RPC: Remote Procedure Call Protocol Specification, Version 2	1057
TELNET Window Size Option	1073
TELNET Terminal Speed Option	1079
TELNET Terminal-Type Option	1091
NFS: Network File System Protocol Specification	1094
TELNET X Display Location Option	1096
DNS Encoding of Network Names and Other Types	1101
U.S. Department of Defense Security Options for the Internet Protocol	1108

Host Extensions for IP Multicasting (STD 5)	1112
Compressing TCP/IP Headers for Low-Speed Serial Links	1144
Structure and Identification of Management Information...(STD 17)	1155
A Simple Network Management Protocol (SNMP) (STD 15)	1157
Line Printer Daemon Protocol	1179
New DNS RR Definitions	1183
Path MTU Discovery	1191
Management Information Base for Network Management...	1213
Tunneling IPX Traffic through IP Networks	1234
BSD Rlogin	1282
The Finger User Information Protocol	1288
Network Time Protocol (Version 3) Specification, Implementation & Analysis	1305
TCP Extension for High Performance	1323
DNS NSAP RRs	1348
Type of Service in the Internet Protocol Suite	1349
The TFTP Protocol (Revision 2) (STD 33)	1350
Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode	1356
TELNET Remote Flow Control Option	1372
Transmission of IP and ARP over FDDI Network (STD 36)	1390

IP Multicast over Token-Ring Local Area Networks	1469
Encoding Header Field for Internet Messages	1505
Applicability Statement for the Implementation of CIDR	1517
An Architecture for IP Address Allocation with DICR	1518
Classless Inter-Domain Routing (CIDR):...Strategy	1519
Dynamic Host Configuration Protocol	1541
Classical IP and ARP over ATM	1577
The Point-to-Point Protocol (PPP) (STD 51)	1661
*Assigned Numbers (*STD 2)	1700
Post Office Protocol - Version 3 (STD 53)	1939
Internet Message Access Protocol - Version 4rev1	2060
Dynamic Host Configuration Protocol	2131
DHCP Options and BOOTPD Vendor Extensions	2132
Dynamic Updates in the Domain Name System (DNS Update)	2136
Secure Domain Name System Dynamic Update	2137
Agent Extensibility (AgentX) Protocol Version 1	2741
Definitions of Managed Objects for Extensible SNMP Agents	2742

Internet, TCP/IP Protocol Suite, and Related Subjects

The following RFCs are also available on more general Internet, TCP/IP, and related subjects:

- RFC 1118, *The Hitchhikers Guide to the Internet*
- RFC 1359, *Connecting to the Internet: What Connecting Institutions Should Anticipate*
- RFC 1392, *Internet Users' Glossary*
- RFC 1432, *Recent Internet Books*
- RFC 1462, *FYI on "What is the Internet?"*
- RFC 1463, *FYI on Introducing the Internet - A Short Bibliography of Introductory Internetworking Readings for the Network Novice*
- RFC 2151, *A Primer on Internet and TCP/IP Tools and Utilities*

The following books are particularly useful references:

- Comer, Douglas E. [1995], *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture, Third edition*, Prentice-Hall.
- Comer, Douglas E. & David L. Stevens [1994], *Internetworking with TCP/IP, Volume II: Design, Implementation, and Internals, Second edition*, Prentice-Hall.
- Comer, Douglas E. & David L. Stevens [1996], *Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications for the BSD Socket Version, Second edition*, Prentice-Hall.
- Frey, Donnalyn, Rick Adams [1989], *A Directory of Electronic Mail, Addressing and Networks*, O'Reilly & Associates, Inc.
- LaQuey, Tracy L. (editor) [1990], *The User's Directory of Computer Networks*, HP Press.
- Perlman, Radia [1992], *Interconnections: Bridges and Routers*, Addison-Wesley.
- Quarterman, John S. [1990], *The Matrix: Computer Networks and Conferencing Systems Worldwide*, HP Press.
- Santifaller, Michael [1991], *TCP/IP and NFS: Internetworking in a UNIX Environment*, translated by Stephen S. Wilson, Addison-Wesley.
- Stallings, William [1991], *Data and Computer Communications, Third edition*, MacMillan.
- Stevens, W. Richard [1990], *UNIX Network Programming*, Prentice-Hall.
- Tanenbaum, Andrew S. [1996], *Computer Networks, Third edition*, Prentice-Hall.
- Tolhurst, William A. et al. [1994], *Using the Internet, Special Edition*, Que Corp.

HPE/VSI Documentation

For details on the OpenVMS operating systems, system services, and utilities, see the appropriate HPE and VSI documentation.

Appendix B. SSH Status Codes

This appendix has tables that show the status codes for the following SSH tools: SSH2, SSH-ADD2, SSH-KEYGEN, SSH-CMPCLIENT, SSH-CERTTOOL, SSH-CERTVIEW, SCP2, and SFTP2.

SSH Client Status Codes

The following table shows the new status codes for the following SSH tools: SSH2, SSH-ADD2, SSH-KEYGEN, SSH-CMPCLIENT, SSH-CERTTOOL and SSH-CERTVIEW clients. These codes are implemented in TCPware 5.0 and higher.

To enable these status code instead of using the pre-TCPware V5.x codes, the logical name TCPWARE_SSH_NEW_STATUS_CODES must be defined system-wide.

Error Code	Error Name	Description
0C1F8044	AGENTBADPASS	Invalid password entered
0C1F804C	AGENTERROR	General error
0C1F806A	AGENTNOAGENT	No agent is available
0C1F8072	AGENTNOFILE	Private key is unreadable
0C1F807A	AGENTNOID	Key not found in authentication agent
0C1F83F1	AGENTOK	Successful operation by agent
0C1F8082	AUTHCANCEL	Authentication cancelled by user
0C1F803C	AUTHFAIL	Authentication failed
0C1F808A	CERT12ENCOD	Certificate PKCS#12 encoding failed
0C1F8092	CERT12SAVE	Failed to save PKCS#12 package

0C1F809A	CERTBADSTATUS	Bad status returned
0C1F80A2	CERTCANTSETPUB	Failed to set public key
0C1F80AA	CERTERROR	Certificate error
0C1F80B2	CERTNO10SIGN	No PKCS#10 requests signed
0C1F80C2	CERTNOSER	No serial number supplied
0C1F80CA	CERTNOVAL12OBJ	No objects to store in PKCS#12 package
0C1F80D2	CERTPRVKEYGEN	Failed to generate private key
0C1F80DA	CERTPRVKEYREAD	Failed to read private key
0C1F80E2	CERTPRVKEYWRT	Failed to write private key
0C1F80EA	CERTUNDEF	Undefined error
0C1F80F2	CERTWRTFILEB64	failed to write base64 file
0C1F80FA	COMPERR	Compression error
0C1F8102	CONNECTFAIL	Connection failed
0C1F80BA	CONNNOTALLOWED	Connection not allowed
0C1F810A	DISCONBYAPP	Session disconnected by application
0C1F8112	E2BIG	Argument list too long
0C1F811A	EABANDONED	Owner cannot release resource
0C1F8122	EACCES	Permission denied
0C1F812A	EADDRINUSE	Address already in use

0C1F8132	EADDRNOTAVAIL	Can't assign requested address
0C1F813A	EAFNOSUPPORT	Address family not supported
0C1F8142	EAGAIN	No more processes
0C1F814A	EALIGN	Alignment error
0C1F8152	EALREADY	Operation already in progress
0C1F815A	EBADCAT	Bad message catalogue format
0C1F8162	EBADF	Bad file number
0C1F816A	EBADMSG	Corrupted message detected
0C1F8172	EBUSY	Mount device busy
0C1F817A	ECANCELED	Operation canceled
0C1F8182	ECHILD	No children
0C1F818A	ECONNABORTED	Software caused connection abort
0C1F8192	ECONNREFUSED	Connection refused
0C1F819A	ECONNRESET	Connection reset by peer
0C1F81A2	EDEADLK	Resource deadlock avoided
0C1F81AA	EDESTADDRREQ	Destination address required
0C1F81B2	EDOM	Math argument
0C1F81BA	EDQUOT	Disk quota exceeded
0C1F81C2	EEXIST	File exists

0C1F81CA	EFAIL	Cannot start operation
0C1F81D2	EFAULT	Bad address
0C1F81DA	EFBIG	File too large
0C1F81E2	EFTYPE	Inappropriate operation for file type
0C1F81EA	EHOSTDOWN	Host is down
0C1F81F2	EHOSTUNREACH	No route to host
0C1F81FA	EIDRM	Identifier removed
0C1F8202	EILSEQ	Illegal byte sequence
0C1F820A	EINPROG	Asynchronous operation in progress
0C1F8212	EINPROGRESS	Operation now in progress
0C1F821A	EINTR	Interrupted system call
0C1F8222	EINVAL	Invalid argument
0C1F822A	EIO	I/O processing error
0C1F8232	EISCONN	Socket is already connected
0C1F823A	EISDIR	Is a directory
0C1F8242	ELOOP	Too many levels of symbolic links
0C1F824A	EMFILE	Too many open files
0C1F8252	EMLINK	Too many links
0C1F825A	EMSGSIZE	Message too long

0C1F8262	ENAMETOOLONG	File name too long
0C1F826A	ENETDOWN	Network is down
0C1F8272	ENETRESET	Network dropped connection on reset
0C1F827A	ENETUNREACH	Network is unreachable
0C1F8282	ENFILE	File table overflow
0C1F828A	ENOBUFS	No buffer space available
0C1F8292	ENODEV	No such device
0C1F829A	ENOENT	No such file or directory
0C1F82A2	ENOEXEC	Exec format error
0C1F82AA	ENOLCK	No locks available
0C1F82B2	ENOMEM	Not enough core
0C1F82BA	ENOMSG	No message of desired type
0C1F82C2	ENOPROTOOPT	Protocol not available
0C1F82CA	ENOSPC	No space left on device
0C1F82D2	ENOSYS	Function not implemented
0C1F82DA	ENOTBLK	Block device required
0C1F82E2	ENOTCONN	Socket is not connected
0C1F82EA	ENOTDIR	Not a directory
0C1F82F2	ENOTEMPTY	Directory not empty

0C1F82FA	ENOTSOCK	Socket operation on non-socket
0C1F8302	ENOTSUP	Function not implemented
0C1F830A	ENOTTY	Not a typewriter
0C1F8312	ENWAIT	No waiting processes
0C1F831A	ENXIO	No such device or address
0C1F8322	EOPNOTSUPP	Operation not supported on socket
0C1F832A	EPERM	Not owner
0C1F8332	EPFNOSUPPORT	Protocol family not supported
0C1F833A	EPIPE	Broken pipe
0C1F8342	EPROCLIM	Too many processes
0C1F834A	EPROTONOSUPPORT	Protocol not supported
0C1F8352	EPROTOTYPE	Protocol wrong type for socket
0C1F835A	ERANGE	Result too large
0C1F8362	EREMOTE	Too many levels of remote in path
0C1F836A	EROFS	Read-only file system
0C1F8372	ESHUTDOWN	Can't send after socket shutdown
0C1F837A	ESOCKTNOSUPPORT	Socket type not supported
0C1F8382	ESPIPE	Illegal seek
0C1F838A	ESRCH	No such process

0C1F8392	ESTALE	Stale NFS file handle
0C1F839A	ETIMEDOUT	Connection timed out
0C1F83A2	ETOOMANYREFS	Too many references: can't splice
0C1F83AA	ETXTBSY	Text file busy
0C1F83B2	EUSERS	Too many users
0C1F83BA	EWOULDBLOCK	Operation would block processing to complete
0C1F83C2	EXDEV	Cross-device link
0C1F8014	EXECERR	Subprocess execution error
0C1F800C	FATALERR	Fatal error
0C1F805A	HOSTNOTALLOW	Host not allowed to connect
0C1F8024	ILLUSER	Illegal username
0C1F801C	KEYEXFAILED	Key exchange failed
0C1F802C	KEYNOTVER	Key not verified
0C1F8034	MACERR	MAC error
0C1F8062	NOMOREMETH	No more authentication methods
0C1F83CA	PROTERR	Protocol error
0C1F83D2	PROTNOTSUP	Protocol not supported
0C1F83DA	SRVNOTAVAIL	Service not available
0C1F83E9	SUCCESS	Successful completion

0C1F8052	TOOMANYCONN	Too many connections
0C1F83E2	UNDEFDISCONCODE	Undefined disconnect reason

SFTP2 Client Status Codes

The following table shows the status codes for the SFTP2 file transfer client.

Error Code	Error Name	Description
0C1F8092	BAD_BUFSIZE	BUFFER_SIZE cannot be less than 512
0C1F809A	BAD_CONCUR	Concurrent_requests requires an argument greater than zero
0C1F807A	BAD_DEBUG	Debug value is out of range
0C1F804A	BAD_DEST	Invalid destination specification
0C1F802A	BAD_PORT_NUM	Port specification is bad or out of range
0C1F8022	BAD_QUALIFIER	Unrecognized command qualifier
0C1F803A	BAD_SOURCE	Invalid source specification
0C1F8082	BAD_TRANSLATE	Bad combination of values for /TRANSLATE_VMS
0C1F800C	CHILD_DIED	SSH2 child process died unexpectedly
0C1F8062	CONNECTION_ERR	Unable to establish or maintain connection to remote system
0C1F805A	DEST_NOT_DIR	Destination is not a directory
0C1F8018	FILE_OVERWRITTEN	Existing file overwritten
0C1F8014	INTERNAL_ERROR	SFTP2 fatal internal error
0C1F8032	MISSING_DEST	Destination file specification is missing
0C1F8072	NO_PERMISSION	Permission denied

0C1F806A	NO_SUCH_FILE	No such file
0C1F8052	PROTO_ERR	Protocol errors
0C1F8042	SOURCE_NOT_AVAIL	Unable to open source file
0C1F80A1	SUCCESS	Successful completion
0C1F808A	TRANSFER_ERR	Error transferring file

SCP2 Client Error Codes

The following table shows the status codes for the SCP2 file transfer client:

Error Code	Error Name	Description
0C1F809A	BAD_BUFSIZE	BUFFER_SIZE cannot be less than 512
0C1F80A2	BAD_CONCUR	Concurrent_requests requires an argument greater than zero
0C1F8082	BAD_DEBUG	Debug value is out of range
0C1F8052	BAD_DEST	Invalid destination specification
0C1F80AA	BAD_OFFSET	Bad offset for READOFFSET or WRITEOFFSET
0C1F8032	BAD_PORT_NUM	Port specification is bad or out of range
0C1F802A	BAD_QUALIFIER	Unrecognized command qualifier
0C1F8042	BAD_SOURCE	Invalid source specification
0C1F808A	BAD_TRANSLATE	Bad combination of values for /TRANSLATE_VMS
0C1F800C	CHILD_DIED	SSH2 child process died unexpectedly
0C1F806A	CONNECTION_ERR	Unable to establish or maintain connection to remote system
0C1F8062	DEST_NOT_DIR	Destination is not a directory
0C1F8018	FILE_OVERWRITTEN	Existing file overwritten

0C1F8014	INTERNAL_ERROR	SCP2 fatal internal error
0C1F803A	MISSING_DEST	Destination file specification is missing
0C1F807A	NO_PERM	Permission denied
0C1F8020	NO_PERMISSION	Permission denied
0C1F8072	NO_SUCH_FILE	No such file
0C1F805A	PROTO_ERR	Protocol errors
0C1F804A	SOURCE_NOT_AVAIL	Unable to open source file
0C1F80B1	SUCCESS	Successful completion
0C1F8092	TRANSFER_ERR	Error transferring file