TCP/IP Applications and Utilities User Guide

HP Part Number: 427639-010
Published: August 2013
Edition: This manual supports J06.03 and subsequent J-series RVUs, H06.03 and subsequent H-series RVUs, and G06.24 and subsequent G-series RVUs

Contents

About This Manual

This manual describes the TCP/IP applications provided for use on HP's implementations of a TCP/IP (Transmission Control Protocol/Internet Protocol) network:

- HP NonStop TCP/IP
- HP Parallel Library TCP/IP

NOTE: Parallel Library TCP/IP is only supported on G-series RVUs.

- HP NonStop TCP/IPv6
- Cluster I/O Protocols (CIP)

NOTE: For this manual, whenever references are made to the TELNET subsystem or process, ECHO, FINGER, FTP, TFTP, and SMTP, the references are to components of the TCP/IP products for HP NonStop systems.

The client and server programs conform to a number of Request for Comments (RFCs). These RFCs define the various Internet protocols implemented by the HP TCP/IP software. Note that conformance to the Internet specifications includes operation of the Internet Protocol over Ethernet networks. For a list of supported protocols and references to the supported RFCs, see the "About this Manual" sections in the following manuals:

- *TCP/IP Configuration and Management Manual*
- *TCP/IP (Parallel Library) Configuration and Management Manual*
- *TCP/IPv6 Configuration and Management Manual*
- *Cluster I/O Protocols (CIP) Configuration and Management Manual*

Supported Release Version Updates (RVUs)

This manual supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, and G06.24 and all subsequent G-series RVUs, until otherwise indicated in a replacement publication.

Intended Audience

This manual is intended for users working interactively at a terminal or workstation. Client applications allow you to access host systems on a TCP/IP network by communicating with server applications on the host systems. This manual also provides information about NonStop server applications that can be accessed through client applications on other systems.

New and Changed Information

Changes to 427639-010 manual:

- Added FTPoption command in Table 5 (page 99).
- Added a section "Runtime Options" (page 116).

New and Changed Information in Previous Editions

The edition of the manual, 427639-009, is revised to add a note for configuring the LISTNER process retries. See "The LISTNER Process" (page 21).

The edition of the manual, 427639-008, is revised to add a reference section for domain name resolution. See "Addressing Remote Hosts" (page 23).

The edition of the manual, 427639-007, is revised to provide the missing definition of FTPerrtext. See "Error Text" (page 105) in "FTP API External Specification" (page 99).

The 427639-006 edition of this manual was been updated to reflect support of a new PARAM that is used to bind LISTNER to an IP address. This feature is supported for J06.09 and subsequent J-series RVUs and H06.20 and subsequent H-series RVUs. It is not supported on G-series RVUs. See "The LISTNER Process" (page 21), Example 1 (page 23), and Example 2 (page 23).

In addition, support of the CIP subsystem has been added throughout the manual, where ever NonStop TCP/IP processes are discussed.

Change bars for this revision have been retained and are shown in the right margin.

Document Organization

Use the table of contents and the index to guide you to the appropriate section or page number. The sections provide the following information:

- Section 1, Introduction to TCP/IP Applications and Utilities, introduces the TCP/IP applications and explains the conventions for addressing remote hosts.
- Section 2, ECHO—Testing Network Connections, describes how to use the ECHO client to test your connection to a remote system.
- Section 3, PING—Checking Host Availability,describes how to use the PING client to test for the availability of a host on the network.
- Section 4, FINGER—Displaying Network User Information, describes how to use the FINGER client to get information about users currently logged on to a system on the network.
- Section 5, Tracer— Tracing IP Packets describes the Tracer Utility Program, which displays the path taken by IP packets on route to a network host.
- Section 6, FTP—Transferring Files, describes how to use the FTP client to transfer files to and from a remote host system.
- Section 7, Communicating With the FTP Server, lists the commands supported by the FTP server and provides a brief overview of the Guardian conventions you use to specify file names and logon information to a NonStop system.
- Section 8, FTP API External Specification, specifies the external interface offered by the FTP application program interface (API).
- Section 9, TFTP—Transferring Public Files, describes how to use the TFTP client to transfer public files to and from a remote host system.
- Section 10, Communicating With the TFTP Server, indicates the level of support provided by the TFTP server and describes file transfer restrictions enforced by the NonStop TFTP server.
- Section 11, TELNET—Using a Network Virtual Terminal, describes how to use the TELNET application to emulate a network virtual terminal to communicate with a remote host on the network.
- Section 12, Communicating With the TELNET Server, explains how the TELNET server establishes a connection when requested through a TELNET client on another system, and lists the TELNET options supported by the server.
- Section 13, Using Your Workstation as a 6530 Terminal, describes how to use the multiple-page terminal emulation utility (TN6530) to emulate a 653X terminal at your workstation.
- Section 14, Addressing Mail to SMTP Hosts, describes how to address mail through the SMTP gateway to users on remote systems in the network.
- Section 15, Anonymous FTP, describes anonymous FTP and how to set it up in both the Guardian and OSS environments.

- Appendix A, Keyboard Mapping for TN6530, describes the keyboard mapping scheme for TN6530.
- Appendix B, Error Messages, provides information on interpreting error messages and correcting the errors.
- Appendix C, Installing TN6530 on a Sun Workstation, provides instructions for installing TN6530 on a Sun workstation.
- Appendix D, TN6530 Control Codes and Escape Sequences, indicates which control and escape sequences are supported by TN6530.
- Appendix E, Parameters for the FTP Server Process in the PORTCONF File, describes optional FTP parameters that can be specified in the PORTCONF file.
- "Abbreviations", defines acronyms and other abbreviations used in this manual and in other NonStop TCP/IP manuals.

Notation Conventions

routine

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS

Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

MAXATTACH

Italic Letters

Italic letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required. For example:

file-name

Computer Type

Computer type letters indicate:

- C and Open System Services (OSS) keywords, commands, and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

  Use the `cextdecs.h` header file.

- Text displayed by the computer. For example:

```
Last Logon: 14 May 2006, 08:02:23
```

- A listing of computer code. For example

```
if (listen(sock, 1) < 0)
{
perror("Listen Error");
exit(-1);
}
```

Bold Text

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

[ ] Brackets

Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name

INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]

K [ X | D ] address
```

{ } Braces

A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }

ALLOWSU { ON | OFF }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis

An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]…

 - ] {0|1|2|3|4|5|6|7|8|9}…
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char…"
```

Punctuation

Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing

Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing

If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]…
```

!i and !o

In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                    !i
                         , error        ) ;              !o
```

!i,o

In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                      !i,o
```

!i:i

In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length           !i:i
                            , filename2:length ) ;       !i:i
```

!o:i

In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                        !i
                        , [ filename:maxlen ] ) ;        !o:i
```

Notation for Messages

This list summarizes the notation conventions for the presentation of displayed messages in this manual.

Bold Text

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

Nonitalic Text

Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

Italic Text

Italic text indicates variable items whose values are displayed or returned. For example:

```
p-register

process-name
```

[ ] Brackets

Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
proc-name trapped [ in SQL | in SQL file system ]
```

{ } Braces

A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
obj-type obj-name state changed to state, caused by
{ Object | Operator | Service }

process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign

A percent sign precedes a number that is not in decimal notation. The % notation precedes an octal number. The %B notation precedes a binary number. The %H notation precedes a hexadecimal number. For example:

```
%005400

%B101111

%H2F

P=%p-register E=%e-register
```

Related Information

The following manuals provide background information about the Guardian application program interface (API) to the NonStop Kernel operating system and supplementary information pertaining to the TCP/IP applications:

- *Guardian Programmer's Guide* contains information about the default terminal attributes.
- *Guardian User's Guide* provides task-oriented instructions for using TACL and basic information about the Guardian API.
- *TACL Reference Manual* provides complete information about TACL commands.
- *TCP/IP Configuration and Management Manual* provides an overview of NonStop TCP/IP and gives instructions for installing, configuring, and managing the NonStop TCP/IP subsystem.
- *TCP/IP (Parallel Library) Configuration and Management Manual* provides an overview of Parallel Library TCP/IP and gives instructions for installing, configuring, and managing the Parallel Library TCP/IP subsystem.
- *TCP/IPv6 Configuration and Management Manual* provides an overview of NonStop TCP/IPv6 and gives instructions for installing, configuring, and managing the NonStop TCP/IPv6 subsystem.

- *Cluster I/O Protocols (CIP) Configuration and Management Manual* provides an overview of CIP and gives instructions for updating, configuring, and managing the CIP subsystem.
- *TCP/IP Programming Manual* describes the programmatic interface to the NonStop TCP/IP software.

Publishing History

Part Number	Product Version	Publication Date
427639-003	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	December 2003
427639-004	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	February 2004
427639-005	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	July 2005
427639-006	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	February 2010
427639-007	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	March 2010
427639-008	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	August 2011
427639-009	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	February 2012
427639-010	TCP/IP Utilities G06, FTP G07, TFTP G06, Telnet G06, SMTP D30	August 2013

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to **docsfeedback@hp.com**.

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

Abbreviations

The following list defines abbreviations and acronyms used in this manual and in other TCP/IP manuals for HP NonStop systems. Both industry-standard terms and HP NonStop terms are included; HP NonStop terms are marked as such. Since this list covers HP NonStop TCP/IP as a whole, not all terms given here are used in this particular manual.

ARP

Address Resolution Protocol

ARPA

Advanced Research Project Agency

BSD

Berkeley System Distribution

CCITT

Consultative Committee for International Telegraph and Telephone

CIP

Cluster I/O Protocols

CSMA/CD
Carrier Sense Multiple Access with Collision Detection
DARPA
Defense Advanced Research Project Agency
DDN
Defense Data Network
DNS
Domain Name Server
DOD
Department of Defense
DSC
Dynamic System Configuration
DSM
Distributed Systems Management
DTE
Data Terminal Equipment
EGP
Exterior Gateway Protocol
EMS
Event Management Service
FTP
File Transfer Protocol
ICMP
Internet Control Message Protocol
IEEE
Institute of Electrical and Electronics Engineers
IEN
Internet Engineering Note
IP
Internet Protocol
ISO
International Standardization Organization
LAN
local area network
LAPB
Link Access Protocol—Balanced
LLC
Logical Link Control
MAC
Media Access Control
MIL-STD
Military Standard

MLAM
Multilan Access Method (now part of TLAM)
NIC
Network Information Center
OSI
Open Systems Interconnection
PDN
Public Data Network
PDU
Protocol Data Unit
PIN
Process Identification Number
RFC
Request for Comments
SAP
Service Access Point
SCF
Subsystem Control Function
SCP
Subsystem Control Point
SMTP
Simple Mail Transfer Protocol
SNAP
Subnetwork Access Protocol
SRI
Stanford Research Institute.
SYSGEN
System Generation Program
TACL
Tandem Advanced Command Language
TAL
Transaction Application Language
TCP
Transmission Control Protocol
TFTP
Trivial File Transfer Protocol
TLAM
Tandem LAN Access Method
TMDS
Tandem Maintenance and Diagnostic System
UDP
User Datagram Protocol

XNS
Xerox Network System
WAN
Wide Area Network
X25AM
X.25 Access Method (HP NonStop term)

1 Introduction to TCP/IP Applications and Utilities

The HP NonStop TCP/IP applications are a set of client and server programs that operate on a NonStop™ system and conform to the protocol family known as TCP/IP (Transmission Control Protocol/Internet Protocol). These client and server programs are supported by all three NonStop implementations of the TCP/IP protocol:

- NonStop TCP/IP
- Parallel Library TCP/IP

  NOTE: H-series systems do not support HP NonStop Parallel Library TCP/IP.

- NonStop TCP/IPv6
- Cluster I/O Protocols (CIP)

(The term TCP/IP is used in this manual to refer to the protocol itself, rather than to a particular HP implementation of it.)

You can use the client programs interactively at your terminal to connect to and communicate with TCP/IP server programs on other machines (such as UNIX systems) in a heterogeneous network. You can also use client programs operating on other machines to connect to and communicate with TCP/IP server programs. It is also possible to use a HP NonStop client to communicate with an HP NonStop server.

In addition to the clients and servers, the TCP/IP applications include the multiple-page terminal emulation utility (TN6530) and a Simple Mail Transfer Protocol (SMTP) gateway through which you can send and receive network mail.

Overview of Applications

The client and server applications provided with TCP/IP are:

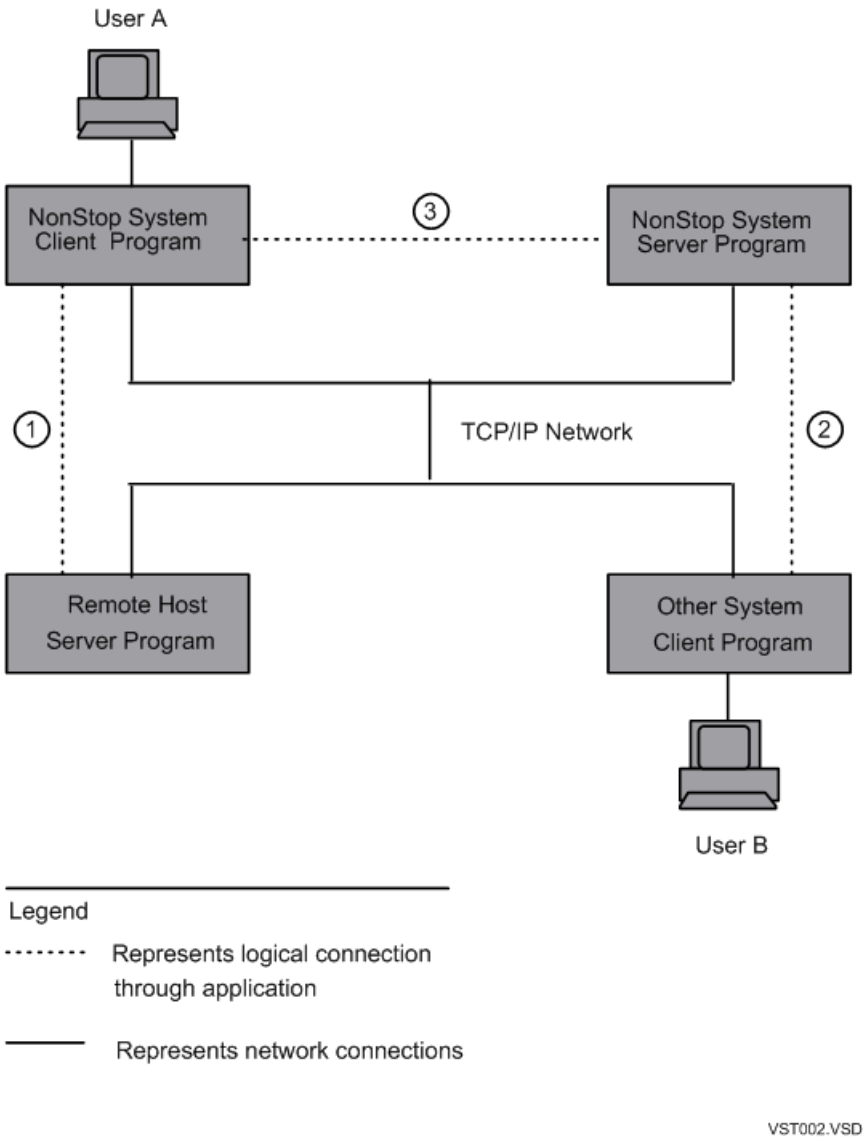
ECHO	You can use the ECHO client to test your connection to a remote system by sending data to the ECHO server on that system. If the connection is successful, the server returns the data you transmitted in the sequence that you entered the data.
PING	You can use the PING client to test whether another host is reachable. PING sends an ICMP echo request message to a host, expecting an ICMP echo reply in return. PING measures the round-trip message exchange time and monitors any packet loss across the network paths.
FINGER	You can use the FINGER client to request information about users that are currently logged on to a system on the network. The type of information and the format of the display depends upon the service provided by the FINGER server on the remote system.
TRACER	You can use the Tracer utility to display the path taken by IP packets on route to a network host.
FTP	You can use the FTP (File Transfer Protocol) client to transfer files to and from a remote system that has an FTP server. In addition, you can choose and display directories on the remote system, delete and rename files, and perform other operations that relate to working with files.
TFTP	You can use the TFTP (Trivial File Transfer Protocol) client to transfer public files to and from a remote system. You can transfer files to or from any system on a network that

	has a TFTP server that accepts requests from the TFTP client. The local and remote systems can place restrictions on which files you can request and where you can store files.
TELNET	You can use the TELNET client to emulate a network virtual terminal connected to a remote host system. You can connect to any system on the network that has a TELNET server. The services available to your terminal depend on what the remote system offers.

The server programs that HP provides for each of these applications allow you to perform similar operations on systems that are made by other vendors and are connected through the network to a NonStop system. You communicate with the NonStop system by invoking the client provided on the foreign system.

Figure 1 illustrates the relation of the clients and servers. In this figure, User A is using a NonStop client program to communicate with a server on a remote host (1). User B is using a client application on some other system to communicate with a NonStop server program (2). User A can also use a NonStop client to communicate with a NonStop server (3) instead of a server made by another vendor.

Figure 1 Using HP NonStop Clients and Servers



TN6530 runs on a UNIX system on a Sun workstation. Using TN6530, you can log on to a NonStop system and run applications that depend on the full capabilities of a terminal in the T16/6530 Multi-Page Terminal family.

You can use the SMTP gateway to send mail to and receive mail from host systems on the internet. You process the mail by using your normal TRANSFER mail service and addressing messages to the SMTP gateway correspondent, which forwards the mail to the remote system.

Using Clients on a NonStop System

To use a NonStop TCP/IP client program, you must be logged on to a NonStop system. You invoke the client program at an operating system prompt (usually, a TACL prompt). Complete instructions for using each client are provided in separate sections of this manual.

Using Servers Through Other Clients

The TCP/IP servers respond to requests for their services sent from a remote system. You must follow the instructions provided with the client that runs on your system.

The ECHO and FINGER servers provide the services specified in the TCP/IP family of protocols. These services are provided over TCP ports, but not over User Data Packet (UDP) ports. The services provided by FTP, TFTP, and TELNET servers are described in Chapter 7 (page 92), Chapter 14 (page 150), and Chapter 12 (page 139).

The LISTNER Process

If you want to use FTP to transfer files into an HP NonStop system, the LISTNER process must be running on that NonStop system. Other aspects of LISTNER functionality are up to you; you can configure the PORTCONF file to have LISTNER support any services you want.

The LISTNER process functions as a super server for the FTP, SMTP, ECHO, and FINGER servers provided by HP. It invokes the appropriate NonStop server as connection requests for FTP, SMTP, ECHO, and FINGER services are received on well-known TCP ports; however, you need not use well-known port numbers for the services. These services do not apply to UDP ports; LISTNER is a TCP-oriented program and listens only to TCP ports. The use of the LISTNER process to invoke several other servers effectively reduces the load on the system.

To use the LISTNER process, configure the $SYSTEM.ZTCPIP.PORTCONF (default) file and start the LISTNER process. If you do not want to use this default file, specify another file by using the RUN LISTNER command. When the LISTNER process is started, it reads from the PORTCONF file to determine which ports it must listen to. The PORTCONF file also defines the servers to be invoked when a request comes in.

NOTE: The LISTNER process reads the PORTCONF file on STARTUP and must be restarted if any changes are made to the PORTCONF file.

Once started, LISTNER reads the SERVICES file to resolve the services configured in the PORTCONF file, and checks that the service name and corresponding port are valid.

NOTE: The PORTCONF file can be shared by two or more LISTNER processes. If the LISTNER processes are running on different TCP/IP processes, TCP/IP port binding succeeds when LISTNER processes are started. If the LISTNER processes use the same TCP/IP process and share the PORTCONF file, then TCP/IP port binding fails when more than one LISTNER process is started.

Consider two LISTNER processes, A and B, which share a PORTCONF file and use the same TCP/IP process. If you attempt to start B after starting A, the TCP/IP port binding fails and returns an EMS message for error EADDRINUSE (4114). The LISTNER processes retry to start after a failure. The number of EMS messages increase with each retry attempt. To avoid creating many EMS messages, you can set the LISTNER^MAXTRY PARAM to limit the number of retries.

For example, the following command sets the retry number to three:

```
PARAM LISTNER^MAXTRY 3
```

The SERVICES file contains information on the known services available on the Internet.

After the accuracy of the PORTCONF file content is verified against the SERVICES file, the LISTNER process listens to the configured ports, waiting for incoming connection requests from the remote client. LISTNER continues to wait for new connections on that port and other well-known ports.

The TCP/IP subsystem (NonStop TCP/IP, Parallel Library TCP/IP, NonStop TCP/IPv6, or Cluster I/O Protocols (CIP)) notifies the LISTNER process that a request is pending. When the LISTNER process receives the notification, it starts the target server.

The target server creates a socket using hostname and source-port information, then accepts the pending connection request on the newly created socket. The TCP/IP subsystem passes a connection request to the LISTNER process only if the port through which the request was received is configured in the PORTCONF file.

Each time a connection is made, connection-oriented services are provided by creating a new LISTNER process. The LISTNER process creates the server process to provide the requested service. The request may be received on a well-known TCP or UDP port.

The LISTNER process then is passed an argument of the form sourceport.sourcehost, where sourceport is a decimal number and sourcehost is a hexadecimal number.

Data can pass between the target NonStop server and the remote client through the newly created socket until either the remote client or the server terminates the connection.

Do not start the LISTNER process from a static or continuously available hometerm. If this condition is not possible specify a local hometerm and IN and OUT files.

NOTE: The LISTNER process must be started by a user who has the user name SUPER.SUPER and the user ID 255,255 in order for a remote FINGER client user to access the information in the DOTPLAN and DOTPROJ files. Otherwise, the FINGER server cannot locate the user's default subvolume in which these files are located.

Parallel Library TCP/IP and NonStop TCP/IPv6 offer a choice of four listening methods. The LISTNER process is used in only one of these methods: the Standard Listening Model. For more information about the Standard Listening Model, see the *TCP/IP (Parallel Library) Configuration and Management Manual* or the *TCP/IPv6 Configuration and Management Manual*. These manuals also contain information about starting the TELNET and TFTP servers.

As of J06.09 and H06.20, you can bind LISTNER to an IP address by using the LISTNER^HOST^IP PARAM. The IP address assigned must be a valid numerical network address, that is, it has to be an IPv4 address in dotted-decimal format or an IPv6 address in hexadecimal format.

To bind LISTNER to an IP address, before starting the LISTNER process, set the TACL PARAM LISTNER^HOST^IP for the IP address. For example:

Example 1 Binding LISTNER to an IPv4 Address

```
PARAM LISTNER^HOST^IP 192.168.10.10
```

LISTNER^HOST^IP also accepts IPv6 addresses. For example:

Example 2 Binding LISTNER to an IPv6 Address

```
PARAM LISTNER^HOST^IP 3ffe:1200:214:1:a00:8eff:fe04:6ef2
```

Addressing Remote Hosts

You can address a remote host by specifying either a host internet address or a host name. Ask the person administering your network for the internet address or name of the host system you want to use.

Using the Domain Name Server (DNS) and IPNODES and HOSTS files

For details about domain name resolution, including the various DEFINEs and PARAMs associated with it, see the *TCP/IP Programming Manual*.

Host Internet Address (IPv4)

NonStop TCP/IP, Parallel Library TCP/IP, NonStop TCP/IPv6, and CIP allow you to specify an IPv4 host address.

A host can have one or more IPv4 addresses on each network to which it is attached. There are three classes of host IPv4 address:

Class A	In class A, the first number is the network address, and the rest of the numbers are the local host address.
Class B	In class B, the first two numbers are the network address, and the rest of the numbers are the local host address.
Class C	In class C, the first three numbers are the network address, and the rest of the numbers are the local host address.

You can also use hexadecimal notation by preceding the hexadecimal digits with 0X or 0x; for example, 0x4f.0x3.0x8.0x16.

Sometimes an IPV4 address is represented externally as two numbers separated by a period: the first number is the network address and the second is the local address; for example, 130.4541.

For examples of various network configurations and detailed information about host internet addresses, see the *TCP/IP Configuration and Management Manual* or the *TCP/IP (Parallel Library) Configuration and Management Manual*.

Host Internet Address (IPv6)

NonStop TCP/IPv6 and CIP allow you to specify an IPv6 host address. There are no classes of IPv6 address.

An IPv6 address contains 128-bits. You represent such an IP address by using a text string in the following format:

```
x:x:x:x:x:x:x:x
```

where x is the hexadecimal value of a 16-bit section of the address. Each of these sections is separated from the others by colons. For example:

```
FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
```

If any 16-bit section contains leading zeros, those zeros need not be entered. For example:

```
1070:0000:0000:0000:0000:0800:200C:417B
```

can be simplified to:

```
1070:0:0:0:0:800:200C:417B
```

When long strings of zeros appear in an address, double colons (::) can be used to represent several 16 -bit sections containing all zeros. For example:

```
1070:0:0:0:0:800:200C:417B
```

can be further simplified to:

```
1070::800:200C:417B
```

The double colon can appear only once in an address. It can, however, be used to represent both leading and trailing zeros.

For examples of various network configurations and detailed information about host internet addresses, see the *TCP/IPv6 Configuration and Management Manual* or *Cluster I/O Protocols (CIP) Configuration and Management Manual*..

Host Name

A host name is the official name by which the host system is known to the Internet. The host name can be associated with the system's Internet address in the name-resolution configuration file or the name can be mapped to an address through a name server. You can ask the system manager of the host system what the host name is. You can also use an alias for a NonStop host if one is defined in the name-resolution file.

Normally, host names are converted to Internet addresses by a Domain Name server. If your network does not have one, host names are resolved through a name-resolution file. For a name-resolution file, you can choose either a HOSTS-type file or an IPNODE-type file. Your choice depends on the NonStop TCP/IP product that you are using and, for NonStop TCP/IPv6 and CIP, whether you are using IPv6 addresses.

HOSTS File

If you are establishing communication by means of IPv4 addresses exclusively, you should use a HOSTS file. If you are running NonStop TCP/IPv6 in DUAL mode, you can use a HOSTS file for resolving the IPv4 addresses.

The HOSTS file is a simple edit type file that contains an entry for each remote host known to your system. Specify each remote host's IP address, host name, and alias. Each entry in the HOSTS file has the following format:

```
IP_address host_name [alias...]
```

The IP_address is a 32-bit numeric value expressed in dotted decimal form. The IP addresses must begin in the first column of an entry in your edit file. The host_name and aliases are alphanumeric and separated by at least one space.

You must configure the DNR to use a HOSTS file; otherwise, DNS is assumed. Use the ADD DEFINE command of TACL to set the TCPIP^HOST^FILE environment variable

For more information about the HOSTS file, see the following manuals:

- *TCP/IP Configuration and Management Manual* (if you are using NonStop TCP/IP)
- *TCP/IP (Parallel Library) Configuration and Management Manual* (if you are using Parallel Library TCP/IP)
- *TCP/IPv6 Configuration and Management Manual* (if you are using NonStop TCP/IPv6)
- *Cluster I/O Protocols (CIP) Configuration and Management Manual*

IPNODES File

For those who use applications in conjunction with NonStop TCP/IPv6, the IPNODES file contains information regarding the known IPv6 (and IPv4) nodes on the network. If you are using INET6 communications and want to use a name-resolution file, you must create $SYSTEM.ZTCPIP.IPNODES

to support local definitions of IPv4 and IPv6 addresses. (For DUAL mode, you can either use HOSTS for IPv4 addresses and IPNODES for IPv6 addresses, or you can put your IPv4 addresses in IPNODES.) Use the ADD DEFINE command of TACL to set the TCPIP^NODE^FILE environment variable.

For more information about the IPNODES file, see the *TCP/IPv6 Configuration and Management Manual* or *Cluster I/O Protocols (CIP) Configuration and Management Manual*.

Resolving File Names

To resolve a TCP/IP file name, use the DEFINE command. The TCP/IP software accesses these names, which it uses to determine the environment in which a program is running.

The following DEFINE names affect the operation of TCP/IP programs:

`=TCPIP^HOST^FILE`	Specifies the name of the HOSTS-type file to be used to resolve names
`=TCPIP^NETWORK^FILE`	Specifies the network addresses and names for `getnetbyaddr` and `getnetbyname` functions
`=TCPIP^PROTOCOL^FILE`	Specifies protocol names and port numbers for `getprotobyname` and `getprotobynumber` functions
`=TCPIP^RESOLVER^NAME`	Specifies the name of the resolver configuration file to be used to get name server information, overriding the default name (`$SYSTEM.ZTCPIP.RESCONF`)
`=TCPIP^SERVICE^FILE`	Specifies service by port number and name for `getservbyname` and `getservbyport` functions
`=TCPIP^PROCESS^NAME`	Specifies the name of the TCP/IP process, overriding the default name (`$ZTC0`). Users of the TFTP server should specify the TCP/IP process by using a TACL PARAM command.

You should ask your system manager if you need to resolve any of these names to run the TCP/IP applications on your system.

For a DEFINE name to affect a running program, the file name must be resolved prior to the execution of the program. When you resolve a file name during an interactive session at a terminal, the specified value stays in effect until you delete it using the DELETE DEFINE command, reset it with the RESET DEFINE command, or log off from the session. You can use the INFO DEFINE command to list the values that are currently set.

See the *TACL Reference Manual* for a detailed description of the various DEFINE commands (ADD DEFINE, INFO DEFINE, RESET DEFINE, and so forth) and their syntax.

See the *TCP/IP Programming Manual* for information about domain name resolution, resolving names with a name server, and using the DEFINE command.

For example, to resolve the `=TCPIP^PROCESS^NAME` name on a system with a TCP/IP process named `$ZTC8`, you would enter the following command before running a client program:

```
TACL 4> ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC8
```

NonStop Kernel Conventions

When transferring files to or from a NonStop Kernel system, you need to know first how to log on, and second, how the system stores disk files and how to specify file names. For complete descriptions of NonStop Kernel conventions, see the *Guardian User's Guide*.

Here is a brief summary of these features:

- To log on to the system, you must provide your user name (or user ID) and your user password. A user name has two parts: your group name and your individual name. You separate the names with a period:

  ```
  MKTG.WILLA
  ```

  You must get a user name from a group manager or system manager of the NonStop system you want to use.

  NOTE: If you do not have a password defined with your user name on a NonStop system, you must use a space when a password is required by a TCP/IP application, such as the FTP client.

- A disk-file name is composed of four parts, separated by periods:

  ```
  \SYS1.$DVOL1.SUBVOLA.FILEZ
  ```

  In this example, \SYS1 is the system name. $DVOL1 is the name of the disk volume where the subvolume that contains the file resides. SUBVOLA is the name of that subvolume, and FILEZ is the name of the file.

  System names begin with a backslash (\) and can contain an additional one to seven alphanumeric characters. You can omit the system name if the file is on the system to which you are logged on. The examples in this manual do not specify a system.

  NOTE: If the backslash is the escape character on your system, enter two backslashes (\\) when specifying a system name.

  Volume names begin with a dollar sign ($) and can contain an additional one to seven alphanumeric characters.

  Subvolume names and file names can contain from one to eight alphanumeric characters and must begin with an alphabetic character.

  If you specify only the file name, the file system assumes the file is on the current default volume and subvolume.

  If you omit the volume name when you specify a subvolume, the file system assumes the subvolume is in the current default volume.

Use of Uppercase

In this manual, Guardian system and subsystem elements such as keywords and file names sometimes appear in uppercase letters, following the convention normally used in manuals for NonStop systems. TCP/IP application commands, keywords, and examples usually appear in lowercase letters, following the UNIX conventions. You can enter information in uppercase or lowercase letters, unless you are specifying a file name in a request to be sent to a remote UNIX system. In that case, you should be aware that the remote system might be sensitive to the case of the letters you enter.

2 ECHO—Testing Network Connections

The ECHO client allows you to test your connection to a remote system by sending data to the ECHO server on that system. If the ECHO server returns the data you transmitted, you know that the server is running and accessible. In other words, the remote server echoes the data you transmit.

For more detailed technical information about ECHO, refer to Network Working Group RFC 862. This RFC is available at several Web sites. To find these sites, type "RFC 862" in the search field of your Internet search engine.

Running ECHO at a Terminal

You can run the ECHO client from any terminal connected to a Guardian system, including a workstation emulating a 6530 terminal or network virtual terminal. HP NonStop ECHO does not service UDP ports.

To run the ECHO client, you enter the ECHO run command. The format of the command is:

```
echo [ / run-option [ , run-option ] ... / ] host
```

run-option

is an operating system RUN command option. See the RUN command in the *TACL Reference Manual* for a complete description of the run options. The most useful run option to use with ECHO is the IN *filename* option. See "Sending a Data File" (page 28), in this section for an example.

host

identifies the remote host system. You can specify *host* as a host name or host address. See Addressing Remote Hosts, in Section 1, Introduction to TCP/IP Applications and Utilities, for information about specifying host names and addresses.

The following command establishes a connection with a remote system named dist101:

```
TACL 4> echo dist101
         Establishing Connection to dist101...Connected
```

When you see the first part of this message, you can assume that ECHO has successfully converted the host name you specified to an internet address. (If you specify a domain name, the data is sent to and received from the Domain Name server.) If you do not see this message, contact your system administrator to report a name resolution problem.

NOTE: Ask your system manager for the name of an appropriate TCP/IP process to serve as your transport service provider.

Considerations for choosing an appropriate process are:

When you see the message "Connected," you know you are connected to the remote system. You will also hear a beep, if your terminal has an audible alarm. The message "Connection Refused" indicates that the remote system is accessible but has no active ECHO server.

Sending One Line of Data

After you have established a connection to the remote system, you can send data by typing a line of characters and pressing the RETURN key. Each line you type is echoed on your terminal screen. If the ECHO server is functioning properly, each line you type should be unchanged.

When you have finished testing the connection, press the CTRL and Y keys simultaneously to disconnect from the remote system.

The following example tests one line of data:

```
TACL 5> echo dist101
     Establishing Connection to dist101...Connected
The quick brown fox jumped over the lazy dog.
```

```
The quick brown fox jumped over the lazy dog.
EOF!
```

Press CTRL/Y

```
TACL 6>
```

Sending a Data File

You can also test the connection by sending a data file. To do this, you first you create a text file containing the lines that you want to send. Then, you specify the file as the IN file in the ECHO run command. For example, assume a file named testdata contains three lines of text. To send the data, enter:

```
TACL 6> echo /in testdata/ dist101
     Establishing Connection to dist101...Connected
Line 1 of test file abcdefghijklmnopqrstuvwxyz
Line 2 of test file 1234567890!@#$%^&*()_-+=
Line 3 of test file |<>>?:",./;'[]~
TACL 7>
```

Reaching the end of file (EOF) causes ECHO to disconnect from the remote system.

3 PING—Checking Host Availability

The PING program is used to test whether another host is reachable. PING sends an Internet Control Message Protocol (ICMP) echo request message to a host, expecting an ICMP echo reply to be returned. PING measures the round-trip time of the message exchange and monitors any packet loss across network paths.

PING program execution is restricted to the SUPER.SUPER user ID.

Running PING at a Terminal

The command syntax for running PING at a terminal is:

```
PING [ -d ] [ -r ] [ -s src-addr ] [ -v ][ -w timeout ]
     host-name [ data-size ] [ num-packets ]
```

[-d]
sets the SO_DEBUG option on the socket being used.

[-r]
bypasses the normal routing tables and sends the message directly to a host on an attached network.

[-s *src-addr*]
specifies that the IP address in *src-addr* should be used as the source address in outgoing probe packets. The address specified in *src-addr* must be an IP address rather than a host name.

On NonStop systems with more than one IP address, you can use the -s option to change the source address to be something other than the IP address that Ping would use by default.

The IP address you specify for *src-addr* must be one of the IP addresses of the NonStop system from which you are sending packets. Otherwise, an error message is returned.

[-v]
specifies verbose output. Results in listing ICMP packets other than ECHO_RESPONS that are received.

[-w]
specifies a waiting period of *timeout* seconds between each packet (sending).

host-name
specifies the host name or host IP address of the target host on the network.

[*data-size*]
specifies the number of data bytes to be sent. The default is 56 which, when combined with the 8-byte ECMP header data, translates to 64 ICMP data bytes.

[*num-packets*]
specifies the maximum number of ECHO_RESPONSE packets to be sent (and received). Command executions stops when this limit is reached.

Examples

The following example shows the entry of the PING command using an IPv4 address. The example includes the result of that command (note that the number of bytes reported includes the 8-byte ICMP header data):

NOTE: Users of all three HP NonStop TCP/IP products (NonStop TCP/IP, Parallel Library TCP/IP, and NonStop TCP/IPv6) can specify IPv4 addresses.

```
> PING 192.168.150.15 80 20

PING 192.168.150.15: 80 data bytes
88 bytes from 192.168.150.15: icmp_seq=0. time=2. ms
88 bytes from 192.168.150.15: icmp_seq=1. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=2. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=3. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=4. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=5. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=6. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=7. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=8. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=9. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=10. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=11. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=12. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=13. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=14. time=2. ms
88 bytes from 192.168.150.15: icmp_seq=15. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=16. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=17. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=18. time=1. ms
88 bytes from 192.168.150.15: icmp_seq=19. time=1. ms

----192.168.150.15 PING Statistics----

20 packets transmitted, 20 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 1/1/2
```

The following example shows the entry of a PING command using an IPv6 address.The example includes the result of that command (note that the number of bytes reported includes the 8-byte ICMPv6 header data):

NOTE: Only users of NonStop TCP/IPv6 and CIP can specify IPv6 addresses.

```
> PING fe80::a00:8eff:fe01:7db8 80 20

PING fe80::a00:8eff:fe01:7db8: 80 data bytes
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=0. time=3. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=1. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=2. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=3. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=4. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=5. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=6. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=7. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=8. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=9. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=10. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=11. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=12. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=13. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=14. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=15. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=16. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=17. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=18. time=2. ms
88 bytes from fe80::a00:8eff:fe01:7db8: icmp_seq=19. time=2. ms

----fe80::a00:8eff:fe01:7db8 PING Statistics----
```

```
20 packets transmitted, 20 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 2/2/3
```

4 FINGER—Displaying Network User Information

The FINGER client allows you to get information about users who are currently logged on to a system on the network. You can request information about a single user or all current users on a system. The type of information provided and the format in which it appears vary depending on what is provided by the FINGER server on the system you specify.

For more detailed technical information about FINGER, refer to Network Working Group RFC1288. This RFC is available at several Web sites. To find these sites, type "RFC 1288" in the search field of your Internet search engine.

Running FINGER at a Terminal

You use the FINGER run command to request information. The format of the command is:

```
finger [/run-option [, run-option ]... /][ user[@host] ]
                                         [ user        ]
                                         [ @host       ]
```

run-option

is an operating system RUN command option. See the RUN command in the *TACL Reference Manual* for a complete description of the run options. The most useful run option to use with FINGER is the OUT *filename* option. See "Sending Information to a File" (page 34), in this section for an example of using this option.

user[@host]

is the name of the user you want described. Specify the user name or user ID as required by the user's system. If you want information about a local user, you can omit @*host*. If the user resides on a Guardian system, *user* must take the form *group.user*, where *group* is the Guardian home group for the *user*.

user

is the name of the user, on the local system, you want described. Specify the user name or user ID as required by the user's system. If the user resides on a Guardian system, *user* must take the form *group.user*, where *group* is the Guardian home group for the *user*.

@*host*

identifies a remote system. You can specify a host name or host address. See Addressing Remote Hosts on page 1-5, in Section 1, Introduction to TCP/IP Applications and Utilities, for information about specifying host names and addresses. If you specify @*host but omit user, information about all users on the host system is displayed*.

If you omit both *user* and @*host*, information about all users on your local system is displayed.

NOTE: Ask your system manager for the name of the TCP/IP process on your system. If the process is named anything other than $ZTC0, you need to resolve the DEFINE name =TCPIP^PROCESS^NAME before running FINGER.

You might also need to resolve the DEFINE name =TCPIP^HOST^FILE.

Refer to the *TCP/IP Configuration and Management Manual* , *TCP/IPv6 Configuration and Management Manual* or the *Cluster I/O Protocols (CIP) Configuration and Management Manual* for more information about DEFINE names, domain name resolution, and starting clients and servers.

Displaying Information

The following examples illustrate the four methods you can use to request information.

All Local Users

To display information about all users currently logged on to your local system, omit the *user* and *@host* parameters.

```
TACL 5> finger
Checking processors...

  Grp     User        Program            TTY     Process Pid
CE.JEAN           $SYSTEM.SYS04.TACL   $AE11     $C191   13,22
DIST.ERNIE        $SYSTEM.SYS04.TACL   $AA40     $C99    15,32
   .                    .                .         .       .
   .                    .                .         .       .
```

Users appear in alphabetic order. The oldest process attached to a single terminal is identified. A backup process is not listed.

All Remote Users

To display information about all users on a remote system, specify the name or address of the remote system. For example, the following command requests information about users of the system named region2:

```
TACL 6> finger @region2

Login       Name          TTY Idle    When     Where
dietrich Lisa Dietrich     pa       Mon 09:30
lozano   Ned Lozano        co 1:26 Wed 09:09  region2
wong     Fred Wong         pb  :32 Fri 15:45  cs8.Tandem.COM
```

The amount of information and the format will vary, depending on which system you contact. This example illustrates typical output from a UNIX system.

A Single Local User

To request information about a single user of your local system, specify *user* only. For example, the following command requests information about the user whose user ID is manufg.sally.

```
TACL 7> finger manufg.sally

  Grp     User        Program            TTY     Process Pid
MANUFG.SALLY      $SYSTEM.SYS04.TACL    $AA21    $C170   13,22
```

As a Guardian system user, you can provide project and plan information by creating two files on your default logon subvolume. See “Providing Information for Other Users” (page 33),

A Single Remote User

To request information about a single user on a remote system, specify *user*@*host*. For example, the following command requests information about a user named kass on a system named salecntr. The information and format are system-dependent:

```
TACL 8> finger kass@salecntr

(Information about user kass appears here.)
```

Providing Information for Other Users

When other users on the network request information about you, FINGER searches for two files on your default logon subvolume. If your subvolume contains an edit file named DOTPROJ, FINGER displays the contents of the file under the heading “Project:”. If your subvolume contains an edit file named DOTPLAN, FINGER displays the information in the file under the heading “Plan:”.

You can create these edit files if you want to communicate your plans and project information. You must secure the files so they can be read by any user on the network.

For example, assume the DOTPROJ file contains the following text:

```
Investigation of feeding habits of the rhinoceros.
```

Assume the DOTPROJ file contains the following text:

```
This project begins January 1990 with two months of field work in Kenya.  The research team will include three
 members who ....
```

A user who requests information about you will be able to determine the type of project you are working on and your plans for its completion.

NOTE: The system will not be able to locate your subvolume unless the FINGER server has been started by someone with super ID capability. See the *Guardian User's Guide* for information about super ID capability.

Sending Information to a File

You can send the information returned in response to your request directly to a file by using the OUT run option. For example, the following command writes information about all remote users of the system named region2 to a file named userinfo:

```
TACL 9> finger /out userinfo/ @region2
```

5 Tracer— Tracing IP Packets

The Tracer Utility displays the path taken by IP packets on route to a network host. Use the Tracer Utility to determine any problems that these packets might encounter. From each gateway system along the path, the Tracer Utility attempts to elicit an ICMP TIME_EXCEEDED message. From the destination remote host, it attempts to elicit a ICMP_PORT_UNREACHABLE message.

Running the Tracer Utility from a Terminal

You can use Tracer Utility only if your user ID is SUPER.SUPER.

Output from the Tracer Utility appears on the screen of the terminal from which the utility was launched. You can also choose to have the output logged to a file.

```
TRACER [ / run-option ] [ , run-option... / ][ -d ]
       [ -m max-ttl ][-n ][ -p port-num ] [ q nqueries ]
       [ -r ][ -s src-addr ] [ -v ] [ -w wait-time ]
       remote-host-name [ data-size ]
```

[run-option]

specifies an operating system RUN command option. For a complete description of all RUN options, see the *TACL Reference Manual*.

Note that the OUT option allows you to send the output of a trace to a log file.

Examples:

The following command directs the output of a trace to remote system named \IDEV to a disk file named $fiti.trace. traceout on the local system.

```
>TACL TRACER/OUT $fiti.trace.traceout/IDEV
```

The following command directs the output of a trace to remote system named \IDEV to a disk file named $wpo.trace. traceout on the system named \igate.

```
>TACL TRACER/OUT \igate.$wpo.trace.traceout/IDEV
```

[-d]

sets the SO_DEBUG option on the socket being used.

[-m max-ttl]

specifies the maximum time-to-live value (or number of hops) used in out-going probe packets. If you do not specify this option, the Tracer Utility uses the default value of 30 hops.

[-n]

specifies that the address of each gateway be printed numerically rather than both symbolically and numerically. Thus, only the IP address, rather than the address and gateway name, appears in the output. Specifying -n avoids having the Tracer Utility do a time-consuming address-to-name lookup. HP therefore recommends use of this option.

[-p *port-num*]

specifies the base UDP port number used in probes. If you do not specify this option, the Tracer utility uses the default value 33434 for *port-num*.

The UDP port number, whether it is the default number or a number you specify through this option, should not be an actual port range on the remote host to which the probe is destined. The remote host should not process the probe packet, but rather send back a

ICMP_UNREACH_PORT message to conclude route tracing. The Tracer Utility informs you of this occurrence by printing an exclamation point either on your screen or in the disk file you specified.

Specifying the -p option is useful when the default value (33434) does specify an actual port range on the destination host. In such cases, -p option allows you to specify an unused port range.

[-q *nqueries*]

specifies the number of probes, or queries, per TTL. If you do not specify this option, the Tracer Utility uses the default value 3 for *nqueries*.

[-r]

specifies that the routing tables be bypassed and that probes should be sent directly to a host on an attached network. If the host is not on a directly attached network, an error message is returned.

You can use this option to check whether a remote host is on an attached network.

[-s *src-addr*]

specifies that the IP address in *src-addr* should be used as the source address in outgoing probe packets. The address specified in *src-addr* must be an IP number rather than a host name.

On NonStop systems with more than one IP address, use the -s option to change the source address to an address that differs from the IP address of the interface on which the probepacket is sent.

The IP address you specify for *src-addr* must be one of the IP addresses of the NonStop system on which you launch the trace. Otherwise, an error message is returned, and the Tracer Utility does not send out any probes.

[-v]

specifies verbose output. If you specify this option, all received ICMP packets are listed. If you do not specify this option, only the ICMP packets TIME_EXCEEDED and UNREACHABLE are listed.

[-w *wait-time*]

specifies the amount of time, in seconds, that the Tracer Utility waits for a response to a probe. If you do not specify this option, the Tracer Utility uses the default value of 5 seconds for *wait-time*.

remote-host-name

specifies the name or IP address of the remote host system to which the Tracer Utility is to trace the path. This parameter is required. You must specify it following any Tracer options (-d, -m, -n, -p, -q, -r, -s, -v, or -w).

[*data-size*]

specifies the packet-size in bytes. If you do not specify *data-size*, the Tracer Utility uses the default probe datagram length. The default probe datagram length is 38 bytes. You must specify *data-size* immediately following the specification of *host-name*.

6 FTP—Transferring Files

The FTP client allows you to transfer files to and from a remote host system while working interactively at your terminal. In addition to transferring files, you can work with directories on the remote system, delete and rename remote files, and use tools such as macros to make your work easier and more efficient.

By using FTP, you can transfer ASCII files, unstructured files, and structured files such as key-sequenced, relative, and entry-sequenced files. Note that transferring SQL files is not supported.

Running FTP at a Terminal

This section tells you how to run the FTP client on your Guardian system to communicate with FTP servers on other remote systems. To use the HP NonStop FTP client commands, you must initiate an FTP session by entering the FTP run command. The form of the FTP run command is:

```
ftp [ / run-option [ , run-option ] ... / ]

    [ -v ] [ -d ] [ -i ] [ -n ] [ -g ] [ -llogfile ] [ -k ] [ –s local IP Address] [ host ]
```

run-option
is an operating system RUN command option. See the RUN command in the *TACL Reference Manual* for a complete description of the run options.

-v
requests FTP to display all responses from the FTP server and report on data transfer statistics; this is called verbose mode.

-d
turns on debugging mode. When debugging mode is enabled, FTP displays each command sent to the remote system and displays the string "-->" preceding each command.

-i
turns off interactive prompting during multiple file transfers. If prompting is enabled, you can selectively retrieve and store files; FTP prompts you for a confirmation before each file is transferred.

-n
prevents FTP from attempting to automatically log you on to the remote host when you initially connect to it, and prevents FTP from prompting for logon information. Unless you include this option, FTP will try to locate a file named FTPCSTM on your logon default subvolume. If the file exists, FTP searches for an entry that describes an account on *host*.

If no entry exists, FTP prompts you for the remote host name or address and, if required, a password and account with which to log on. See "Logging On to a Remote System" for more information.

-g
turns off the FTP mechanism for expanding a single name to a list of file names based on the wild-card characters included in the name. See the `glob` command description for detailed information.

- *llogfile*
logs the output of FTP to the log file specified in *logfile*. You can specify the name of a disk file, spooler job, or EMS collector for *logfile*.

If you specify an EMS collector, HP recommends that you specify a collector other than $0 to avoid flooding $0 with FTP messages.

-*s local IP Address*

turns on the feature of specifying a local IP address for the FTP client.

-*k*

turns on the feature of sending TCP *Keep-Alive* probes on the client side control connection.

host

identifies the remote host system. You can specify *host* as a host name or host internet address. See Addressing Remote Hosts on page 1-5 in Section 1, Introduction to TCP/IP Applications and Utilities, for information about specifying host names and addresses.

If you do not specify *host*, you must use the `open` command to connect to a remote host system.

The following command starts FTP, turns on verbose mode, turns off interactive prompting, and establishes a connection to a host system named warehs1:

```
TACL 5> ftp -v -i warehs1
```

NOTE: Ask your system manager for the name of an appropriate TCP/IP process to serve as your transport service provider.

Considerations for choosing an appropriate process are:

After you enter the FTP command to start an FTP session and, optionally, log on to a remote host, FTP displays the prompt:

```
ftp>
```

You can enter any of the FTP client commands to perform file operations.

In addition to specifying options when you start FTP, you can enter them at the ftp> prompt. Each option except -n has a corresponding command that also turns the toggle on or off. The options and corresponding toggle commands are:

```
-v    verbose
-d    debug
-i    prompt
-g    glob
```

Table 1 summarizes the FTP commands organized by function. A detailed description of each command is given in the upcoming section, "FTP Command Reference".

Table 1 FTP Command Summary

Command	Function	Type	G/OSS
!	Escape to shell		OSS
$	Execute macro	Session Tools	G/OSS
?	Request local help information	Session Tools	G/OSS
account	Supply remote account password	Connections to Remote Systems	OSS
append	Append local file to remote file	File Transfer Operation	G/OSS
ascii	Set ASCII transfer type	File Transfer Parameters	G/OSS
aslinemode	Allows FTP to receive lines longer than 239 characters—user can specify cut, wrap, or none (do nothing) to long ascii lines	File Transfer Parameters	G

Table 1 FTP Command Summary *(continued)*

Command	Function	Type	G/OSS
bell	Turn ringing of bell after file transfer on or off	Session Controls	G/OSS
binary	Set binary transfer type	File Transfer Parameters	G/OSS
bye	Disconnect from remote system and exit FTP	Connections to Remote Systems	G/OSS
case	Turn case mapping of remote file names on or off during execution of get or mget commands	File Naming Control	G/OSS
cd	Change remote working directory	Remote Directories	G/OSS
cdup	Change to parent directory of remote working directory	Remote Directories	G/OSS
contimer	Sets the value for the length of time the client waits for an incoming connection request from the server.	Session Control	G
close	Disconnect but do not exit from FTP.	Connections to Remote Systems	G/OSS
cr	Toggle carriage return stripping on ascii gets		G/OSS
debug	Turn debugging mode on or off.	Session Controls	G/OSS
delete	Delete remote file	File Management Operation	G/OSS
dir	Display detailed listing of remote working directory	Remote Directories	G/OSS
disconnect	Disconnect but do not exit from FTP	Connections to Remote Systems	G/OSS
exit	Disconnect from remote system and exit FTP	Connections to Remote Systems	G
filecode	Specify file code for newly created local binary files	File Transfer Parameters	G
form	Set file transfer format	File Transfer Parameters	G/OSS
get	Copy remote file to local subvolume.	File Transfer Operation	G/OSS
glob	Turn wild-card name expansion on or off	Session Controls	G/OSS
hash	Turn display of hash mark (#) during file transfer on or off	Session Controls	G/OSS
help	List local help information	Session Tools	G/OSS
lcd	Change current local subvolume	Local Subvolumes	G/OSS
ls	Display brief listing of remote working directory	Remote Directories	G/OSS
macdef	Define macro of FTP commands	Session Tools	G/OSS

Table 1 FTP Command Summary *(continued)*

Command	Function	Type	G/OSS
mdelete	Delete multiple remote files	File Management Operation	G/OSS
mdir	Display information about multiple remote files/directories.	Remote Directories	G/OSS
mget	Copy multiple remote files to local subvolume	File Transfer Operation	G/OSS
mkdir	Create remote directory	Remote Directories	G/OSS
mls	Display brief information about multiple remote files/directories	Remote Directories	G/OSS
mode	Set file transfer mode	File Transfer Parameters	G/OSS
mput	Copy multiple local files to remote directory	File Transfer Operation	G/OSS
nlist	List contents of remote directory	Remote Directories	OSS
nmap	Define or turn off mapping scheme for file names	File Naming Control	G/OSS
ntrans	Define or turn off translating scheme for file names	File Naming Control	G/OSS
open	Connect to remote system	Connections to Remote Systems	G/OSS
prompt	Turn prompting on or off for confirmation before file operation	Session Controls	G/OSS
proxy	Execute FTP command on secondary control connection bad	Connections to Remote Systems	G/OSS
put	Copy local file to remote directory	File Transfer Operation	G/OSS
pwd	Display name of remote working directory	Remote Directories	G/OSS
quit	Disconnect from remote system and exit FTP	Connections to Remote Systems	G/OSS
quote	Send specific arguments verbatim	Session Controls	G/OSS
recv	Copy remote file to local subvolume	File Transfer Operation	G/OSS
remotehelp	Request remote help information	Session Tools	G/OSS
rename	Rename remote file	File Management Operation	G/OSS
reset	Clear reply queue to synchronize with remote system	Session Tools	G/OSS
rmdir	Remove remote directory	Remote Directories	G/OSS
runique	Turn mechanism for assigning unique name to	File Naming Control	G/OSS

Table 1 FTP Command Summary *(continued)*

Command	Function	Type	G/OSS
	local copy of remote file on or off		
send	Copy local file to remote directory	File Transfer Operation	G/OSS
sendport	Turn use of port command for each data connection on or off	Session Controls	G/OSS
site	Precedes all site-specific commands. Table 7-2 in Site-Specific Commands on page 7-4 describes these site-specific commands.	Remote Session Controls/Tools	G/OSS
status	Display information about your FTP session	Session Tools	G/OSS
struct or stru	Set file transfer structure	File Transfer Parameters	G
sunique	Turn on or off the mechanism for assigning unique name to remote copy of local file	File Naming Control	G/OSS
tenex	Set data representation type for TENEX machines	File Transfer Parameters	G/OSS
trace	Toggle packet tracing		G/OSS
type	Display or set data representation type	File Transfer Parameters	G/OSS
user	Supply remote user name (or user ID)	Connections to Remote Systems	G/OSS
verbose	Turn on or off the display or responses from remote FTP server	Session Controls	G/OSS

Connecting to a Remote System

To establish a connection to a remote system, you can either include the host name or address in the FTP run command, or you can enter an open command at the ftp> prompt; for example, the following command connects to a system named warehs1:

```
ftp> open warehs1
```

Before you connect to another system (unless you are using the proxy command), you must close the current session by using either the `close` or `disconnect` command.

Logging On to a Remote System

There are three ways to log on to a remote system:

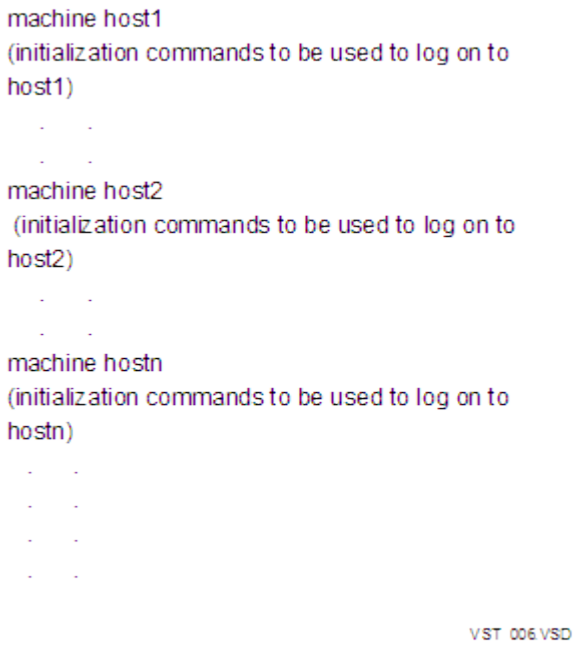
- You can disable automatic logon and logon prompting by including -n in the FTP run command and using the user command to log on after you start FTP.
- You can create an entry in the FTPCSTM file that FTP can use to perform the logon operation automatically either when you start FTP (and specify a host) or when you enter an open command. If you use this method, you cannot include -n in the FTP run command.
- If your logon default subvolume does not contain an FTPCSTM file or the FTPCSTM file does not contain an entry for the *host* you specify when you connect to a remote system, FTP prompts you for logon information. In this case, you must also omit -n from the FTP run command.

Creating an FTPCSTM Entry

The FTPCSTM file contains logon and initialization information used when automatic logon is turned on. The subvolume on which you create this file must be your logon default subvolume when you start FTP or enter an open command.

The file named FTPCSTM can be an edit file in the format shown in Figure 2.

Figure 2 FTPCSTM File Format



You can include the following initialization commands, separated by spaces or tabs, or placed on separate lines.

`machine` *`host`*

The following FTPCSTM file contains logon information for two systems—`hosta` and `hostb`:

```
machine hosta
login troy
password helen
macdef st
status

machine hostb
login troy
password clef
```

When you enter an open hosta command, you are automatically logged on as user `troy`, and the macro named st is defined. For another example of an `FTPCSTM` file, see the macdef command in the "FTP Command Reference" (page 48).

FTP Default Personality Selection

Currently, the FTP default personality selection criteria is based on whether or not the INITIAL-DIRECTORY attribute in the user's authentication record is set. This attribute can be viewed using the SAFECOM command:

`INFO USER` *`user-id`*`, DETAIL`

If a valid initial directory is set, the user will be logged on with a NonStop™ Kernel Open System Services (OSS) default personality. If no initial directory is set, the user will be logged on with a

Guardian personality. However, if OSS is not running, or an invalid or non-existent initial directory is set, the FTP code allows the normal user to be logged on with the Guardian personality.

NOTE: For the anonymous user, if an invalid or non-existent INITIAL-DIRECTORY is provided or if OSS is not running, the anonymous user logon will be denied.

Logging On by Responding to Prompts

The following example illustrates how FTP prompts you for information when you omit the -n option, but no FTPCSTM file entry exists for the remote system:

```
ftp> open warehs1
Connecting to warehs1.zzzco.com....Established
Name (warehs1:guest): troy
Password: zombie
```

You must follow the conventions of the system to which you are connected when entering your user name and password. FTP requires that you supply a password when logging on. If you are logging on to a NonStop system and do not have a password defined on that system, you can enter a space for the password.

Logging On with the User Command

If you specify the `-n` option when you start FTP, you must use the user command to log on; for example, the next command specifies a user named `troy`, whose password is `zombie`, and an whose account is named `sales`:

```
ftp> user troy zombie sales
```

Logging Off and Stopping an FTP Session

To log off a remote system but to continue your local FTP session, use either the close or disconnect command:

```
ftp> close
```

You can then use the open command to log on to another remote system.

If you want to stop your FTP session and at the same time log off the remote system, use either the `bye` or `quit` command:

```
ftp> bye
```

Stopping a File Transfer Operation

To interrupt a file transfer operation, press the BREAK key. At the command interpreter prompt, use the STOP command to abort the FTP process:

```
ftp> put newaccts

Press BREAK key.

TACL 6> stop
```

Controlling FTP Interaction

FTP provides several ways you can control your interaction with the system. You can use toggle commands to turn mechanisms such as the ringing of a bell on or off. In addition to the toggle commands, you can use the reset command to clear the reply queue when a remote server violates the FTP protocol.

You can use the status command to display the current setting of toggle commands and other FTP session controls.

The following example illustrates how to turn on the bell and the debugging mode before logging on to a remote host:

```
ftp> bell
ftp> debug
ftp> open warehs1
```

Setting File Transfer Parameters

You can set values for several file transfer parameters that affect the way in which file transfers take place. In addition, you can assign a Guardian file code to a binary file created by either the `get` or `recv` command by using the `filecode` command. The default file code for binary-mode transfers is 0.

Each file transfer parameter has a default value when you start FTP. To change a parameter's value, you use the FTP command with the same name as the parameter. For example, use the mode command to change the file transfer mode, and use the `type` command to change the data representation type. (You can also use the `ascii`, `binary`, and `tenex` commands to change the current data representation type.) The commands that you use to set these parameters are summarized in Table 1.

In the next example, the binary and filecode commands set the current data representation type to binary and the filecode to 888:

```
ftp> binary
ftp> filecode 888
```

You can also change the data representation type with the following command:

```
ftp> type binary
```

When you use either the put or send command to send a file to a Guardian system on which the file does not already exist, the FTP server creates the file with an eight-page primary extent and, at most, 977 sixteen-page secondary extents. When you are transferring large files, you can improve performance by specifying larger extent sizes. This allows the system to allocate fewer extents while the file is being copied. See the description of either the put or send command for information about how to specify extents.

Using Directories

FTP provides several commands that you can use to work with directories on the remote system.

To determine the name of the current working directory on the remote system, use the pwd command; to change to the parent directory of the current one, use the cdup command. In the following example, assume a UNIX directory structure consists of the /usr directory that contains a directory named /tmp.

```
ftp> pwd
257 "/usr/tmp" is current default.
ftp> cdup
ftp> pwd
257 "/usr" is current default.
```

You can use the cd command to select a specific directory. You can also use the mkdir command to make directories, and the `rmdir` command to remove directories.

Displaying Information

You can use the `dir`, `ls`, `mdir`, and `mls` commands to display information from directories on a remote system.

The following commands display the name of the current directory, the contents of the directory named projects, and abbreviated information about all files in the current directory with have

names beginning with the lowercase letter g (for more information, see "Specifying a Set of Files" (page 45)),

```
ftp> pwd
257 "/usr" is current default.
ftp> dir projects

(A list of file names and file attributes appears.)

ftp> mls g*

(A list of file names beginning with g appears.)
```

When you request directory information, you can specify that you want the output to be sent to a file instead of displayed on your terminal. For example, the following command writes the list of files from the projects directory to a file named `dirinfo`:

```
ftp> ls projects dirinfo
```

Specifying File Names

In many FTP commands, you specify the names of local and remote files. To specify a local file, you must follow the Guardian file naming conventions. To indicate that the file to receive data is the standard input file (usually your home terminal), specify a hyphen (-). For example, this command displays a remote file named `memo825` on your terminal:

```
ftp> get memo825 -
```

To specify a remote file name, you must know the naming conventions used by the remote system.

When entering a HP NonStop fully qualified file name (that is, one that contains a system node name), remember that many systems use the backslash character as an escape character. In such cases, you must enter the fully qualified file name by using two backslash characters. For example:

```
ftp> GET \\SYS1.$VOL1.MSS.Messages
```

When specified with a parameter of `\\sys.$vol.subvol.filename`, the MPUT, and PUT commands will transfer the specified file to the remote system, where it will be given the name `\sys.$vol.subvol.filename`. For MGET or GET commands, FTP will retrieve the remote file named `\sys.$vol.subvol.filename`.

Access to Guardian Spoolers

get and put commands support access to Guardian spoolers. For example, to send the remote file report to a the Guardian spooler `$S.#titanu`, issue the following command:

```
ftp> get report $S.#titanu
```

To send the local file analysis to the remote Guardian spooler `$S.#comprt`, issue the following command:

```
ftp> put analysis $S.#comprt
```

Specifying a Set of Files

When you copy or delete multiple files, you can specify the set of files with a wild-card name (a name that contains wild-card characters). The local or remote system expands the wild-card name (global name) to a list of files. This technique is called "globbing."

When you want to specify a wild-card name, you must make sure that the glob toggle is on. You can use the glob command to turn globbing on and off. To see the current setting of glob, you can use the status command:

```
ftp> glob
ftp> status
Connected to warehs1.zzzco.COM.
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
```

```
Verbose: on; Bell: off; Prompting: on; Globbing: on
       .
       .
```

When glob is on, a wild-card name in an mput, mget, or mdelete command is expanded to a list of file names. For the mget and mdelete commands, the remote system expands the name and creates the list. For the mput command, the local system performs this task.

The rules for using wild-card characters and examples of their use are given in the description of the glob command in the "FTP Command Reference" (page 48).

Mapping and Translating File Names

Because the local system and the remote system usually have different file-naming conventions, FTP provides commands you can use to define file-name mapping and translating schemes. For example, when you copy a file to a remote system and do not specify a name for the file, FTP uses the local file name after applying the current translating and mapping schemes you have defined.

The commands that define the mapping scheme and the order in which these commands are applied to a name are: case, ntrans, and nmap. If you want a file name used without changes, you can turn case off (the default) and reset ntrans and nmap by entering these commands without any parameters.

For descriptions of these commands, see the "FTP Command Reference" (page 48).

The following example illustrates the effect of combining a mapping scheme with a translating scheme:

```
ftp> ntrans +- pm
ftp> nmap $1/$2 $2
ftp> mget maindir/x*
```

When you issue the mget command, the remote system expands the wild-card name to all files in the directory named maindir which begin with the letter x. The asterisk indicates that the letter x can be followed by one or more characters of any type. FTP creates the local names for these files by first performing the translation and then the mapping. A plus sign (+) is translated to the letter p and a minus sign (-) to the letter m. The first slash character and the part of the name preceding it are deleted when the name is mapped, as in the following example:

```
Remote File Name    Translated Name   Mapped Name

maindir/xyz+abc     maindir/xyzpabc   xyzpabc

                       |       |         |
                      $1      $2        $2
```

For details on filename mapping and translation, see the nmap and ntrans commands in the "FTP Command Reference" (page 48).

The following mapping scheme maps UNIX names to Guardian names:

```
ftp> nmap /usr/tmp/$1 $vol1.subvol.$1
```

In this example, a file named /usr/tmp/abc on a UNIX system is mapped to a file named `$vol1.subvol.abc` on a Guardian system.

Sending Data

FTP allows you to send copies of one or more files to a remote system or send specific arguments to the system.

Sending Copies of Local Files

You can send a copy of a local file to a remote system by using either the put or send command, or you can send copies of multiple files by using the mput command.

Either of the following commands sends a copy of a file named `newaccts` on the current default subvolume on the local system to the current working directory on the remote system. The name of the remote file will also be newaccts, modified by the current settings of nmap and ntrans:

```
ftp> put newaccts
ftp> send newaccts
```

If you specify a remote file name, FTP sends that name to the remote system exactly as you specify it. You can specify a simple file name or a name qualified by directory names, such as a UNIX pathname.

To send copies of multiple files, you use the mput command and specify a wild-card name for the set of local files you want to send. The following command copies to the remote system all files that have names ending with the four letters `acct`, followed by a single character:

```
ftp> mput *acct?
```

The local system expands the wild-card name entry `*acct?` to a list of files that match the pattern. Each of the local file names is modified according to the current setting of ntrans and nmap to obtain the name of the corresponding remote file.

If the prompt toggle is on, FTP prompts you to confirm the transfer of each file, as in the following example:

```
ftp> mput *acct?
mput BIGACCT1? y
200 PORT command successful.
        .
        .
mput NEWACCT3? n
mput OLDACCT3? y
        .
```

Sending Specific Arguments

The quote command sends arguments you specify directly to the FTP server on the remote system. This command is provided to allow you to bypass normal command parsing to accommodate a special situation. You should use this command carefully, because sending an invalid command could disrupt your current FTP session.

The following quote command sends the four letters noop, exactly as they are entered:

```
ftp> quote noop
```

Receiving Data

FTP allows you to receive copies of one or more files from a remote system. You can copy a remote file to your local system by using either the `get` or `recv` command, or you can make copies of multiple remote files by using the mget command.

Either of the following commands copies a remote file named `delinqac` from the current working directory on the remote system to the current default subvolume. The name of the new file will also be `delinqac` modified by the current settings of nmap and `ntrans`:

```
ftp> get delinqac
ftp> recv delinqac
```

You can specify a name for the local file, as in the following example:

```
ftp> get delinqac dlqacct
```

To make copies of multiple remote files, you use the mget command and specify a wild-card name for the set of remote files you want to copy. The following command copies all files that have names beginning with the three letters `sal`:

```
ftp> mget sal*
```

The remote system expands the wild-card name entry `sal*` to a list of files that match the pattern. The local name for each file consists of the remote file name, modified according to the current settings of `case`, `ntrans`, and `nmap`.

If you want to be prompted before each file is copied, set the prompt toggle on.

Deleting Files

You can delete a single remote file by using the `delete` command, or a set of files by using the `mdelete` command with a wild-card name. For example, the following command deletes all files from the current working directory that begin with the letter `z` and contain exactly three additional characters. Assume the glob toggle is on:

```
ftp> mdelete z???
```

Each question-mark matches any character, but each must match at least one. For a description of the wild-card characters, see the `glob` command in the "FTP Command Reference" (page 48).

To delete a file named customer from the directory named accounts, enter the following commands:

```
ftp> cd accounts
ftp> delete customer
```

If you want to be prompted before each file is deleted, set the prompt toggle on.

FTP Anonymous User Support

See Chapter 15 (page 152), for information about FTP anonymous user support.

FTP Command Reference

The following pages contain descriptions of the syntax (enclosed in boxes) and rules for using FTP commands, and provide examples of typical ways to use the commands.

account

Use the account command to supply a supplemental password required for access to resources on a remote system. You can use the account command after you successfully log on to the remote system.

```
account [ password ]
```

password

is the password for the remote account. To prevent the password characters from being displayed on the screen, omit *password*. The system then prompts you for the password and does not echo the characters you enter.

If the remote system does not have accounts, a message appears indicating that the account command is not implemented on that system.

Examples

In the following example, the account password `zinger` is sent to the remote system:

```
ftp> account zinger
```

In the next example, the system prompts for the password. The password is not displayed as you type it:

```
ftp> account
Account:
ftp>
```

The password prompt varies from system to system.

append

Use the `append` command to append the contents of a local file to a file on a remote system.

```
append local-file [ remote-file ]
```

`local-file`

is the Guardian file name of the file to be appended.

`remote-file`

is the name of the remote file. Specify *`remote-file`* as required by the remote system.

If you do not specify *`remote-file`*, FTP uses the *`local-file`* name, after altering the name based on the current settings of `ntrans` and `nmap`.

FTP uses the current settings of `type`, `form`, `mode`, and `stru` to determine how to append the file.

When applied to a key-sequenced file, the append command appends only a record that does not already exist in the file. If the record does exist, FTP returns an error message.

Example

To append a local file named newrules to a remote file named `guidelines`, enter:

```
ftp> append newrules guidelines
```

ascii

Use the ascii command to set the data transfer type to ASCII. Use the ASCII transfer type when you want to transfer ASCII files (files containing printable ASCII characters only), such as text files or spooler files, or to transfer structured file types, such as key-sequenced, entry-sequenced, and relative ENSCRIBE files. The ASCII transfer type is the default value when you start an FTP client session. You must have a connection established with an FTP server (called an FTP session) before you can issue the ascii command. When using the ASCII data transfer, tab characters in the file being transferred are changed to blanks. See the description of the `binary` command if you want to transfer a binary file. The get and put commands describe how to transfer files.

```
ascii
```

Example

To change the data transfer type to ASCII, enter:

```
ftp> ascii
```

aslinemode

Use the aslinemode command to receive files with lines longer than 239 ASCII characters. Depending on the aslinemode command option specified, FTP will either truncate lines longer than 239 characters, insert a line feed character after the 239th character, or abort after receiving the 239th character in a line.

```
aslinemode { none | cut | wrap }
```

none

causes the FTP receive to abort after receiving the 240th ASCII character in a line. This is the default.

cut

FTP truncates each line in a received file to 239 ASCII characters.

wrap
: FTP inserts a line feed character after receiving the 239th ASCII character in a line. This is the only way to ensure that you will receive all the data in a file with lines longer than 239 characters

Example

For incoming files, to cause FTP to ignore all characters after the 239th in each line, type:

```
ftp> aslinemode cut
```

bell

Use the bell command to turn on (or off) a mechanism to sound a bell after each file transfer completes. This command operates as a toggle.

```
bell
```

Example

To change the setting of the bell toggle, enter:

```
ftp> bell
Bell mode on.
```

binary

Use the binary command to set the data transfer type to binary. Use the binary transfer type when you want to transfer binary files, such as program files. Since the ASCII transfer type is the default value when you start an FTP client session, you will have to explicitly enter the binary command before transferring binary files. You must have a connection established with an FTP server (called an FTP session) before you can issue the binary command. See the description of the ascii command if you want to transfer ASCII files or structured files. The get and put commands describe how to transfer files.

```
binary
```

Example

To change the data transfer type to binary, enter:

```
ftp> binary
```

bye

Use the `bye` command to disconnect from the FTP server and exit FTP.

```
bye
```

Example

To disconnect from a remote system and exit FTP, enter the bye command:

```
ftp> bye
TACL 7>
```

case

Use the `case` command to turn on (or off) case mapping of remote file names during an operation initiated by a `get` or `mget` command. This command operates as a toggle.

```
case
```

When you start an FTP session, case is off. If you transfer a file to the local system, the uppercase letters in the remote file name are not changed to lowercase letters.

If you turn case on, remote file names having all uppercase letters are stored on the subvolume with lowercase letters.

Because case is not significant in Guardian names, you do not need to use this command. It is included in this implementation for compatibility with other versions of FTP.

Example

To change the setting of case, enter the following:

```
ftp> case
Case mapping on.
```

cd

Use the cd command to change the current working directory on the remote system.

```
cd remote-directory
```

remote-directory
is the name of the remote directory you want to use.

Example

The following command changes from the current directory to the accounts directory:

```
ftp> cd accounts
```

cdup

Use the cdup command to change the working directory on the remote system to its parent.

```
cdup
```

Example

Assume a directory named nwaccts resides in a directory named sales. If `nwaccts` is the current working directory, the following command changes the working directory to `sales`:

```
ftp> cdup
```

close

Use the `close` command to disconnect from a remote server and return to the FTP client prompt.

```
close
```

The close command erases any currently defined macros.

Example

The following commands open and close a connection to the remote system named `warehs1`:

```
ftp> open warehs1
        .
        .
ftp> close
ftp>
```

contimer

Use the contimer command to set the timeout interval for accepting the connection from the server.

```
contimer timeout
```

timeout
is the time in seconds. Specify an integer value in the range 0 through 32767.

If the connection timeout value is not set through contimer command, the FTP client waits indefinitely to accept the connection from the FTP server.

Example:

To set a connection timeout on the client side for 50 seconds, use the following command.

```
ftp> contimer 50
```

In the above example, FTP client would wait for 50 seconds to accept the connection from the server before returning the prompt back to the user.

cr

Turns carriage return stripping on or off during ascii file-type retrieval. Records are denoted by a Carriage Return/Line Feed sequence during ascii file-type transfers. When cr is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single-linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; in this case, when an ascii file-type transfer is made, these linefeeds can be distinguished from a record delimiter only if cr is off.

```
cr { on | off }
```

on

Turns carriage return stripping on during ascii file-type retrieval. This is the default setting.

off

Turns carriage return stripping off during ascii file-type retrieval.

debug

Use the `debug` command to turn debugging mode on or off. This command can be used as a toggle or used to set an explicit mode.

```
debug [ debug-value ]
```

`debug-value`

is an integer that specifies the setting of debugging mode explicitly. Zero turns debugging off; any other value turns debugging on. If you do not specify *`debug-value`*, the command performs as a toggle and the current setting is reversed.

When debugging mode is on, FTP displays each command sent to the remote system; the command is preceded by the characters --->.

Example

Assume debugging mode is off. The following example illustrates the result of turning debugging mode on:

```
ftp> debug
Debugging on (debug=1).
ftp> mput zt*
mput ZTST123?y
---> PORT 127,0,0,1,4,54
---> STOR ZTST123
mput ZTX?
      .
      .
      .
```

delete

Use the delete command to delete a file from a remote system.

```
delete remote-file
```

`remote-file`

is the name of the file you want to delete. Specify *`remote-file`* as required by the remote system.

Example

To delete a file named `custlist`, enter the following command:

```
ftp> delete custlist
```

dir

Use the dir command to display the contents of a directory on the remote system.

```
dir [ remote-directory ] [ local-file ]
```

remote-directory

specifies the name of the directory you want to display. Specify *remote-directory* as required by the remote system. If the remote host is a NonStop system, specify this in the form *subvolume.**.

NOTE: HP NonStop does not support the Big File feature on the OSS platform. Thus, an OSS user cannot use the dir command to determine the size of a file exceeding 2 GB. An attempt to use the dir command for such a file displays a 0 in the EOF column. To see the actual size of the file, an OSS user must issue the dir command in the Guardian mode. For a discussion of the Big File feature, see the *File Utility Program (FUP) Reference Manual*.

If you omit *remote-directory*, the current working directory is displayed.

local-file

specifies a local file to which the output is sent. If the file does not exist, FTP creates a text file.

If you omit *local-file*, the information appears on the current IN file, usually your home terminal.

If you want to display the information in a brief format, use the `ls` command.

Example

The first command in this example requests the content of the `sales` directory to be displayed on the terminal. The second command sends the same content to a local file named salesout.

```
ftp> dir sales

(Directory information appears here.)

ftp> dir sales salesout
```

disconnect

Use the `disconnect` command to disconnect from a remote server and return to the FTP prompt.

```
disconnect
```

The `disconnect` command performs the same operation as the `close` command.

Example

The following commands connect to and then disconnect from the system named warehs1.

```
ftp> open warehs1
        .
        .
ftp> disconnect
ftp>
```

exit

Use the `exit` command to disconnect from the FTP server and exit FTP.

```
exit
```

Example

To disconnect from a remote system and exit FTP, enter the following command:

```
ftp> exit
TACL 7>
```

filecode

Use the `filecode` command to specify a file code to be assigned to a newly created file during a `get` or `recv` file transfer operation. The code is applied only when the current data representation type is binary. A file code is an attribute of a Guardian file.

`filecode` *`code`*

`code`
is an integer from 0 through 32767. The default code is 0.

Example

To set the default file code to 100, enter the following command:

```
ftp> filecode 100
Binary filecode set to 100
```

form

Use the form command to set the current format. The default format is non-print. (This is the only format currently available with NonStop FTP.)

`form` *`format`*

`format`
is the name of the format; only non-print is valid.

With non-print format, printer formatting codes are not interpreted. This format is normally used with files that are going to be processed or stored.

Example

The following command sets the format:

```
ftp> form non-print
```

get

Use the get command to copy a remote file to the local system. You must have a connection established with an FTP server (called an FTP session) before you can issue the `get` command. The file transfer is based on the current status of all FTP options, toggles, and macros. Use the FTP `status` command to display these values. If the file you wish to copy is a structured file containing embedded `<CRLF>` character sequences, you must first issue the stru command with the r option. See the `stru` command for more details.

The get command differs in the way it handles extents for structured files and unstructured files.

If you issue a `get` command for an unstructured file and a file with the same name already exists on the local system, the command purges the old file and creates a new file with default extents. However, if you specify extents in the `get` command by using the *`primary`*, *`secondary`*, and *`maxextents`* parameters, the command creates a new file containing the specified extents.

If you issue a `get` command for a structured file and a file with the same name already exists on the local system, the command does not purge the old file, but merely overwrites it. The extents of the old file are retained in the new file. Thus, any extents you specify are ignored.

To support the transfer of large files, the FTP Client and Server alters the line numbering scheme. Files with fewer than 40,000 records are numbered sequentially. Records ranging from 40, 000 through 340,000 are given the delta increment of .1. Records greater than 340,000 are given the delta increment of .001.

The syntax for transferring ASCII or binary files is:

```
get remote-file [ local-file[ , attribute-list ]]


where attribute-list is:


[ [filecode],[primary],[secondary],[maxextents] ]
```

The syntax for transferring structured files is:

```
get remote-file [ local-file[ , attribute-list ]]


where attribute-list is:


[[filetype],[filecode],[primary],[secondary],[maxextents],
 [record-len],[pri-key-len],[key-offset],[index-blk-len ]]
```

remote-file

specifies the name of the remote file to be copied. Specify *remote-file* as required by the remote system.

local-file

specifies a name for the local file. If you omit *local-file*, FTP uses the name of the *remote file* for the new file name, after modifying the name according to the current settings of the case, ntrans, and nmap commands. If you specify a hyphen (-) as the local file, the file will be displayed on the screen only (not locally saved).

attribute-list

is a list of file attributes for a HP NonStop file. You can specify these attributes only if the local file system is a Guardian.

You must specify the attributes in the order indicated, using commas as place holders and omitting spaces. You must not include a space after the comma that separates the *local-file* specification from any of the file attributes, such as *filecode*, *pext*,and so on. Issue the `FUP INFO` *remote-file*, `DETAIL` command to obtain detailed information about the file to be copied.

filecode

indicates the file code of the local file. *filecode* is a number from 0 through 32767. When FTP is in binary transfer mode, the default file code is 0. When FTP is in ASCII transfer mode, the default file code is 101. This attribute will override the current setting of the `filecode` command.

primary

indicates the primary extent size in pages (2048-byte units) of the local file. *primary* is an integer from 1 through 65535. The default is determined internally.

secondary

indicates the secondary extent size in pages (2048-byte units) of the local file. *secondary* is an integer from 1 through 65535. For structured files, the default size is 16 pages. For unstructured files, the default is determined internally.

maxextents

indicates the maximum number of extents of the local file. *maxextents* is an integer from 1 through 978. The default value is 978 extents. When your command specifies a *maxextents* value lower than 16 but greater than 0, FTP sets the *maxextents* value to 16.

NOTE: When you are transferring a large file, you can improve performance by specifying larger extent sizes allowing the system to allocate fewer extents while the file is being copied.

filetype

for structured file transfer, indicates the type of file:

e	indicates an entry-sequenced file
k	indicates a key-sequenced file
r	indicates a relative file

record-len

indicates the length of the records in a structured file. You must obtain this information using the FUP INFO command with the DETAIL option (see the Note below).

pri-key-len

indicates the primary key length in a structured file. You must obtain this information using the FUP INFO command with the DETAIL option (see the Note below).

key-offset

indicates the key offset in a structured file. You must obtain this information using the FUP INFO command with the DETAIL option (see the Note below).

index-blk-len

indicates the index block length in a structured file. You must obtain this information using the FUP INFO command with the DETAIL option (see the Note below).

NOTE: To ensure that structured files are correctly transferred, you must first issue the FUP INFO command with the DETAIL option for the file you wish to transfer. All file attributes that are present in the display must be included in the get command. The following example shows the file attributes for a file named KEYSEQ:

Examples

Example 1

This example copies a remote ASCII file, named BIGEDIT, to the local system.

First, before starting FTP, obtain a detailed listing of the file:

```
$55> fup info $ABC.source.bigedit,detail

$ABC.SOURCE.BIGEDIT                    19 May 1995,  9:16
    ENSCRIBE
    TYPE U
    CODE 101
    EXT ( 8 PAGES, 16 PAGES )
    ODDUNSTR
    MAXEXTENTS 978
    BUFFERSIZE 4096
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  10 Oct 1994,  8:01
    CREATION DATE:  10 Oct 1994,  8:00
```

```
    LAST OPEN:  19 May 1995,  9:05
    EOF: 775804 (2.4% USED)
    FILE LABEL: 250 (6.1% USED)
    EXTENTS ALLOCATED: 25
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP put command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> get $abc.source.bigedit bigedit,101,8,16,978
200 PORT command successful.
150 Opening data connection for $abc.dest.source bigedit(133.255.17.1) (775804 bytes).
226 Transfer complete.
local: bigedit remote: $abc.source.bigedit
746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Example 2

This example copies a remote, key-sequenced, structured file, named `KEYSEQ`, to the local NonStop system

First, before starting FTP, obtain a detailed listing of the file:

```
$6> fup info keyseq,detail

$ABC.SOURCE.KEYSEQ                    19 May 1995,  9:17
    ENSCRIBE
    TYPE K
    CODE 1001
    EXT ( 5 PAGES, 5 PAGES )
    REC 60
    BLOCK  512
    IBLOCK 512
    KEYLEN 4
    KEYOFF 0
    MAXEXTENTS 16
    OWNER 165,86
    SECURITY (RWEP): NUNU
    DATA MODIF:  10 Oct 1994, 16:13
    CREATION DATE:  10 Oct 1994, 16:13
    LAST OPEN:  19 May 1995,  9:05
    EOF: 8704 (5.3% USED)
    FILE LABEL: 214 (5.2% USED)
    EXTENTS ALLOCATED: 1
    INDEX LEVELS: 1
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP put command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
```

```
230 User MANUALS.WRITER logged in.
ftp> get keyseq source.keyseq,k,1001,5,5,16,60,4,0,512
200 PORT command successful.
150 Opening data connection for source.keyseq (133.255.17.1) (7680 bytes).
226 Transfer complete.
local: keyseq remote: source.keyseq
6200 bytes sent in  0.10 seconds (60.55 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Example 3

This example copies a remote ASCII file named `remofile` to a file named `data4a` on the local system. The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, you should use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the CODE, EXT, and MAXEXTENTS values are the same as those for BIGEDIT in Example 1.

At the TACL prompt, start FTP and open a connection to the FTP server using the IPv6 address. Then, using the information from the FUP INFO display, issue the FTP `get` command:

```
\IDC15.$FITI.FTPMARCH 90> ftp  fe80::a00:8eff:fe01:7db8
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8......Established.
220 idc15.om.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.data
250 CWD command successful.
ftp> get remofile, data4a 101,8,16,978
200 EPRT command successful.
150 Opening data connection for remofile
(fe80::a00:8eff:fe01:7db8,1347d).
226 Transfer complete.
Local: data4a remote: remofile
746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Example 4

This example copies a remote, key-sequenced, structured file named `keyseq1` to a file named `keyseq2` on the local NonStop system, The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the TYPE, CODE, REC, IBLOCK, KEYLEN, KEYOFF, and MAXEXTENTS values are the same as those for the file in Example 2.

Next, start FTP and open a connection to the desired FTP server using its IPv6 address. Then, using the information from the FUP INFO command, issue the FTP get command:

```
\IDC15.$FITI.FTPMARCH 91> ftp  fe80::a00:8eff:fe01:7db8
\IDC15.$FITI.FTPMARCH 91..
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS
INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8.........Established.
220 idc15.wipro.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
```

```
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.remote
250 CWD command successful.
ftp> lcd $fiti.q9552d
Local working vol.subvol:  "\IDC15.$FITI.Q9552D"
ftp> get keyseq1 keyseq2,k,1001,5,5,16,60,4,0,512
200 EPRT command successful.
150 Opening data connection for keyseq (fe80::a00:8eff:fe01:7db8,1349d).
226 Transfer complete.
local: keyseq2 remote: keyseq1
6200 bytes sent in  0.10 seconds (60.55 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

glob

Use the `glob` command to turn the mechanism for expanding wild-card names on or off. This command operates as a toggle. The glob toggle applies only to `mdelete`, `mget`, and `mput` commands. Wild-card names you specify in an `mls` or `mdir` command are always expanded.

`glob`

A wild-card name contains wild-card characters. FTP uses the wild-card characters to expand the name to a list of file names. When glob is off, wild-card names are interpreted literally and are not expanded. When glob is on, you can specify a set of files in mdelete, mget, and mput commands by using a wild-card name.

When you specify a wild-card name for local files in an mput command, the wild-card characters are processed as shown in Table 2. The wild-card name is replaced with an alphabetically sorted list of file names that match the pattern. If you want a name that includes a period to be selected, you must include the period (.) explicitly in the pattern; for example, a*.b* matches all.bits, but a*b* does not.

A wild-card name you specify in an `mdelete` or `mget` command is expanded on the remote system. The result of expanding a wild-card name depends on the name processing done by the operating system and FTP server on the remote system.

NOTE: If you enter more than one wild-card name, the lists that result are not merged; the same file name might appear in more than one list.

Table 2 Wild-Card Characters for Local File Names

Characters	Meaning
*	Matches any string of characters including the null string; for example, z* matches the names zoo, zombie, zy122, and z.
?	Matches any single character; for example, z? matches the names zx and zm, but not the names z or zoo.
[c...c]	Matches any one of the characters (c) enclosed; for example, a[bcd]e matches the names abe, ace, and ade. You can use a hyphen (-) between two characters to indicate an alphabetic range; for example, [a-ei-k] is a short way to specify [abcdeijk].
~	Indicates your logon volume and subvolume when specified at the beginning of the pattern; for example, ~.z* selects all files that begin with the letter z on your logon default

Table 2 Wild-Card Characters for Local File Names *(continued)*

Characters	Meaning
	volume and subvolume ($*logon-volume*.*logon-subvol*.z*).
{ s,...s }	Matches any one of the strings (s) enclosed; for example, fil{old,new} matches the names filold and filnew. The left to right order of resulting patterns is preserved so that file names that match each resulting pattern are grouped in the resulting list. You can nest lists of strings; for example, a{b, c{de,fg}} is the same as a{b,cde,cfg} and matches the names ab, acde, acfg.

Example

The following commands turn glob off, copy a file named `cust?nm` from a remote system, and then turn glob on. The question mark is a normal character in the new local file named cust?nm.

```
ftp> glob
Globbing off.
ftp> mget cust?nm
ftp> glob
Globbing on.
```

hash

Use the hash command to turn on or off a mechanism for displaying a hash mark (#), sometimes called a pound sign, after each data block is transferred. The size of a data block is 4096 bytes. This command operates as a toggle.

```
hash
```

When you start an FTP session, hash is off.

Example

The following example shows the result of turning hash on:

```
ftp> hash
Hash mark printing on (4096 bytes/hash mark).
ftp> get custom1
#
ftp>
```

help

Use the help command to display either a list of all commands or helpful information about a specific command.

```
help [ command-name ]
```

command-name

specifies an FTP command. If you omit *command-name*, a list of all FTP commands is displayed.

Example

The following command requests information about the `cdup` command.

```
ftp> help cdup
cdup   change remote working directory to parent directory
```

lcd

Use the lcd command to change the current subvolume on the local system.

```
lcd [ subvolume ]
```

subvolume

is the Guardian name of the subvolume you want to use. If you omit *subvolume*, your default subvolume becomes the current one.

Example

To change to a subvolume named $vol1.memos, use the following command:

```
ftp> lcd $vol1.memos
Local working vol.subvol: "$vol1.memos"
```

ls

Use the ls command to display an abbreviated listing of the contents of a directory on a remote system.

```
ls [ remote-directory ] [ local-file ]
```

remote-directory

specifies the name of the remote directory. Specify *remote-directory* as required by the remote system. If the remote host is a NonStop system, specify the name in the form *subvolume*.*.

If you omit *remote-directory*, the current working directory is displayed.

local-file

directs the output to the local file that you specify. If the file does not exist, FTP creates a text file. If you omit *local-file* or if you specify a hyphen (-) as *local-file*, the output appears on your terminal.

Example

The following command displays on the terminal the contents of the `docs` directory.

```
ftp> ls docs
```

macdef

Use the macdef command to define a macro of FTP commands.

```
macdef macro-name
```

macro-name

assigns a name to the macro. The name can contain from one through eight characters. Any character except a space or tab is acceptable. If you define a macro named init (in FTPCSTM), the macro is automatically executed as the last step of the logon process.

After you enter the macdef command, enter the set of FTP commands that define the macro—one command per line. To terminate the definition, enter consecutive RETURNs at your terminal (or a blank line in a text file).

Once defined, a macro remains defined until you enter a close or disconnect command.

You can have at most 16 macros defined at one time; the total number of characters in all defined macros cannot exceed 4096.

To invoke a macro, enter $*macro-name* followed by any arguments the macro requires. If you omit the arguments, FTP prompts you for them.

You can use the following special characters in a macro definition:

$n

Macros can save time, particularly if you include the definitions in the `FTPCSTM` file, which executes during automatic logon.

Examples

In the following example, the macro getf is defined and then executed. This macro specifies a mapping scheme and copies a file from a remote system:

```
ftp> macdef getf
Enter macro line by line, terminating it with a null line
nmap $vol1.subvol.\$1 \$1
get $1
                                Press RETURN to end macro. ftp> $getf filexyz           Execute the macro for
filexyz. nmap $vol1.subvol.$1 $1
get filexyz
    .
    .
ftp>
```

Note that you must precede each `$1` in the nmap command with a backslash to indicate that `$1` is an argument of nmap and not an argument of the macro itself. When FTP processes the macro, the backslash characters are stripped before the commands are executed. In the get command, `$1` is an argument of the macro; the value for the argument in this example is `filexyz`.

A macro that sets FTP session options (toggles) and file transfer parameters might be useful. For example, assume the FTPCSTM file contains the following lines:

```
machine salesa Specify logon commands for login troy SALESA host. password zing
account zingzing
macdef setup Define macro named setup. binary Make type binary. case Override default case setting. nmap 1/2/3
  3 Define mapping scheme. ntrans %^@  pc Define translation scheme. prompt Override default prompt setting.
runique Override default runique setting. pwd Display remote working directory.
macdef po Define macro named po. proxy open $1 $2 Include macro variables in proxy open.
```

When you automatically log on to the `salesa` system, the macros `setup` and `po` are defined.

To execute the macro po, enter the remote system name (`$1`) and, optionally, the port number (`$2`). For example:

```
ftp> $po salesb
```

The next example defines the macro `putfiles`. This macro can be used to copy more than one file to a remote system:

```
ftp> macdef putfiles
Enter macro line by line, terminating it with a null line
put $i
 Press RETURN to end macro. ftp> verbose on
ftp> $putfiles a1 b1 c1 Execute the macro for files put a1    named a1, b1, and c1. 200 PORT command successful.
150 Opening data connection for a1...
226 Transfer complete.
local: a1 remote a1
310 bytes sent in 0.01 seconds...
put b1
200 PORT command successful.
   .
   .
put c1
   .
   .
local: c1 remote c1
4200 bytes sent in 0.15 seconds
```

mdelete

Use the `mdelete` command to delete a set of files from the remote system.

`mdelete` *`remote-files`* [*`remote-files`*] ...

`remote-files`

is a file name or a wild-card name specifying the files to be deleted. If you specify a wild-card name, the glob toggle must be on.

Example

Assume the prompt toggle is on. The following command deletes from the current working directory on the remote system all files that begin with the four letters cust and all files that end with the letter x:

```
ftp> mdelete cust* *x
mdelete CUST123?y
mdelete CUST451?n
mdelete ACCT40X?y
```

.
.

mdir

Use the `mdir` command to display directory information about one or more remote files.

```
mdir remote-files [ remote-files ] ... local-file
```

remote-files
: is a file name or a wild-card name that specifies the remote files you want described.

local-file
: sends the output to a local file instead of your terminal. FTP prompts you to confirm that you want to send output to the specified file. If the file does not exist, FTP creates a text file.

  If you want to display the output, specify a hyphen (-) as the local file.

  This is a required parameter. You must specify either a file name or a hyphen.

Examples

To display information about all files in the current working directory, enter the following command:

```
ftp> mdir *
(local-file) -

(Directory information appears here.)
```

To direct the output to a local file named ldir, enter the following mdir command and respond to the prompt by typing the letter y:

```
ftp> mdir * ldir
output to local-file: ldir? y
ftp>
```

mget

Use the mget command to copy one or more remote files to the local current default subvolume.

```
mget remote-files [ remote-files ] ...
```

remote-files
: is a file name or a wild-card name that specifies the remote files you want to copy. If you specify a wild-card name, the glob toggle must be on.

You can use the lcd command to change the default subvolume before you use mget.

To generate local names for the copied files, FTP applies the current settings of case, ntrans, and nmap to the remote file names.

Example

The following command copies all files that begin with the seven letters payroll, followed by two more characters.

```
ftp> mget payroll??
mget PAYROLL45? y
mget PAYROLL58? y
    .
    .
```

mkdir

Use the mkdir command to create a directory on the remote system.

```
mkdir directory-name
```

directory-name

specifies the name of the new directory. Specify *directory-name* as required by the remote system.

The directory is created in the current working directory.

Example

To create a directory named memos, use the following command:

```
ftp> mkdir memos
```

mls

Use the mls command to display abbreviated directory information about one or more remote files.

```
mls remote-files [ remote-files ] ... local-file
```

remote-files

is a file name or a wild-card name that specifies the remote files you want described.

local-file

sends the output to a local file. FTP prompts you to confirm that you want to send output to the specified file. If the file does not exist, FTP creates a text file.

If you specify *local-file* by typing a hyphen (-), the output appears on your terminal.

This is a required parameter. You must specify either a file name or a hyphen.

Example

Assume glob is on. The following command displays the file names cust11, cust45, and cust99:

```
ftp> mls cust{11,45,99}

cust11
cust45
cust99
```

mode

Use the mode command to set the transmission mode. The default mode is stream. (This is the only mode currently available with NonStop FTP.)

```
mode [ mode-name ]
```

mode-name

specifies the transmission mode. Only stream is valid.

In stream mode, data is transmitted as a stream of bytes.

Example

To set the mode, enter the following command:

```
ftp> mode stream
```

mput

Use the mput command to copy one or more local files to the current working directory on the remote system.

```
mput local-files [ local-files ] ...
```

local-files

is a file name or a wild-card name that specifies the local files you want to copy. If you specify a wild-card name, the glob toggle must be on. If you do not qualify a file name, the file is taken from the current default subvolume. The file name is case-sensitive.

To generate remote names for the copied files, FTP applies the current settings of ntrans and nmap to the local file names.

NOTE: When you transfer multiple binary files to a remote T16 system using mput, the file code for all destination files is 0.

Example

The following command copies the local files neword and `oldord` to the remote system:

```
ftp> mput {new,old}ord
mput NEWORD?y
mput OLDORD?y
```

nlist

Use the nlist command to obtain a list of a remote directory.

```
nlist
```

Example

```
ftp> nlist

(Directory information appears here.)
```

nmap

Use the nmap command to define or turn off a mechanism for mapping local file names to remote file names during mput, `put`, and append operations and for mapping remote file names to local file names during mget and `get` operations.

```
nmap [ inpattern outpattern ]
```

`inpattern`

defines a template that applies to the name of the original file (the local file in put and append operations and the remote file in get operations).

To turn the mapping mechanism off, omit both *`inpattern`* and *`outpattern`*.

`outpattern`

defines a template that determines the resulting name for the new file (the remote file in put and append operations and the local file in get operations).

If you specify a target name for a get or put operation, the nmap scheme is not applied.

To define a mapping scheme, first use the following conventions to specify *`inpattern`* as a template for incoming file names:

`$n`

Next, specify *`outpattern`* using the following conventions:

`$n`

The current settings of `case` and `ntrans` are applied to the incoming file name before the name is mapped.

Example

In this example, the mapping scheme moves characters that follow a plus sign (+) before the plus sign and deletes letters that precede the plus sign. For example, file name one+two is mapped to `two+`.

```
ftp> nmap $1+$2 $2+
```

See "Mapping and Translating File Names", earlier in this section for more examples.

ntrans

Use the ntrans command to define or turn off a mechanism for translating local file names to remote file names during mput, put, and append operations and for translating remote file names to local file names during mget and get operations.

```
ntrans [ inchars [ outchars ] ]
```

inchars

is a sequence of characters to be translated; each character is translated to the character in the corresponding position in *outchars*; for example, if *inchars* is abc and *outchars* is xyz, FTP translates the name of the file being copied by changing the letter a to x, the letter b to y, and the letter c to z.

If a character in *inchars* does not have a corresponding character in *outchars*, the character is deleted from the file name; for example, if the value of *inchars* is the letters abc and the value of *outchars* is xy, the letter c is deleted from any file name being translated.

To turn the translating mechanism off, omit *inchars* and *outchars*.

outchars

specifies the characters to which *inchars* characters are translated.

If you specify a target name for a get, put, or append operation, the ntrans scheme is not applied.

If case is on, it is applied before ntrans.

Examples

The following command defines a translation scheme in which the letter x translates to the number 1, the letter y translates to the number 2, and the letter z is deleted.

```
ftp> ntrans xyz 12
```

When translating files from a UNIX system to a NonStop system, the following translation applies: UNIX allows in file names special characters that the Guardian operating system does not allow:

```
ftp> ntrans +-_!%=:/
```

You might want to use a longer list of characters for the value of *inchars*. All the *inchars* characters will be deleted from the incoming file name.

After you finish transferring the files to which this scheme applies, you can reset as follows:

```
ftp> ntrans
```

open

Use the open command to connect to the FTP server on a remote system.

```
open host [ port ]
```

host

is a host name or host address identifying the remote system. See “Addressing Remote Hosts” in Section 1, Introduction to TCP/IP Applications and Utilities.

port

specifies the number of a port on the remote system that you want FTP to contact. If you omit *port*, the well-known (default) port is 21.

If you did not specify the -n option when you initiated your FTP session, FTP attempts to log you on to the FTP server.For more information, see “Running FTP at a Terminal” (page 37).

Example

To connect to the remote system named `sales3` through port 23, enter the following command:

```
ftp> open sales3 23
Name (sales3:guest): user
Password:
ftp>
```

If you do not have a password on the remote system, enter a space and press the RETURN key.

prompt

Use the prompt command to turn interactive prompting on or off. If prompt is on, you can selectively retrieve, store, and delete files; FTP prompts you before each file is transferred. This command operates as a toggle.

`prompt`

If prompt is off, mget and mput operations transfer all files in the expanded list, and mdelete operations delete all files in the expanded list.

When you start an FTP session, prompt is on.

Example

Assume prompt is off. The following command shows the result of turning prompt on:

```
ftp> prompt
ftp> mput abc*
mput ABCDE? y
      .
      .
      .
mput ABCXY? n
ftp>
```

proxy

Use the proxy command to execute an FTP command on a secondary connection. You can use the proxy command to connect to a second remote FTP server (secondary) while still connected to the current FTP server (primary). Using the local FTP client, you can issue commands to transfer files between the two remote servers.

`proxy` *`ftp-command`*

`ftp-command`

specifies the FTP command that you want to execute.

Assume you are already connected to a primary FTP server. You can follow these guidelines to also connect to and communicate with a secondary FTP server:

- First establish a connection to the secondary FTP server by using the *`proxy open`* command. Your FTP client is then connected to both FTP servers.
- You can use the regular FTP commands to get information about or transfer data between the primary FTP server and the local host.
- To get information about files and directories on the secondary FTP server, use the proxy commands; for example, `proxy dir`, `proxy ls`, and `proxy pwd`.
- If the FTP servers also support the `PASV` command, you can transfer data between the two remote servers by issuing proxy commands. (You can use the `remotehelp` command to check whether the server supports the `PASV` command.)

  The `proxy get` and `proxy mget` commands copy files from the primary to the secondary server.

The proxy put, proxy mput, and proxy append commands copy files from the secondary to the primary server.

NOTE: The proxy put and the proxy append commands will take filecode, primary, secondary, and maxextents for remote filename.

Examples

Assume you are working at a terminal connected to the local system in the main office. This system is named main. If you want to copy a file from sales district A's system (salesa) to sales district B's system (salesb), you first log on to one of the systems.

Assume you have already started FTP and you enter open to log on to salesa as your primary host. Then, you enter a proxy open command to log on to the secondary host, salesb.

```
ftp> open salesa
Connecting to salesa.zzzco.COM....Established.
220 comm......
Name (salesa.guest): sales.joan
331 Password required .....
Password:

230 User SALES.JOAN logged in.
ftp> verbose off
Verbose mode off.
ftp> proxy open salesb
Name (salesb:guest): salacct.joanr
Password:
ftp> dir

(Information about files on the primary server is displayed.)
ftp> proxy dir

(Information about files on the secondary server is displayed.)
```

To copy a file from salesa to salesb you use the proxy get command. Specify the file named filea on salesa as remote-file and the file named *fileb* on salesb as local-file.

```
ftp> proxy get filea fileb
local: fileb remote: filea
```

To close the secondary connection (to salesb), enter:

```
ftp> proxy close
```

You are still connected to the salesa system.

put

Use the put command to copy a local file to the remote system. You must have a connection established with an FTP server (called an FTP session) before you can issue the put command. The file transfer is based on the current status of all FTP options, toggles, and macros. Use the FTP status command to display these values. If the file you wish to copy is a structured file containing imbedded <CRLF> character sequences, you must first issue the stru command with the r option. See the stru command for more details.

The put command differs in the way it handles extents for structured files and unstructured files.

If you issue a get command for an unstructured file and a file with the same name already exists on the remote system, the command purges the old file and creates a new file with default extents. However, if you specify extents in the put command by using the *primary*, *secondary*, and *maxextents* parameters, the command creates a new file containing the specified extents.

If you issue a put command for a structured file and a file with the same name already exists on the remote system, the command does not purge the old file, but merely overwrites it. The extents of the old file are retained in the new file. Thus, any extents you specify are ignored.

The syntax for transferring ASCII or binary files is:

```
put local-file [ remote-file ] [ ,attribute-list ]

where attribute-list is:

[ [filecode],[primary],[secondary],[maxextents] ]
```

The syntax for transferring structured files is:

```
put local-file [ remote-file ] [ ,attribute-list ]

where attribute-list is:

[[filetype],[filecode],[primary],[secondary],[maxextents],
  [record-len],[pri-key-len],[key-offset],[index-blk-len ]]
```

local-file
: specifies the name of the local file to be copied. The local file must be an Enscribe disk file. It cannot be a temporary file, a running process, or a Spooler location.

remote-file
: specifies the name for the remote file or remote spooler. If you specify a remote spooler, the data representation type has to be set to ASCII. If you omit *remote-file*, FTP uses the name of the `local file` for the new file name after modifying the name according to the current settings of the `case`, `ntrans` and nmap commands.

attribute-list
: is a list of file attributes for a HP NonStop file. You can specify these attributes only if the remote system is a NonStop system.

  You must specify the attributes in the order indicated, using commas as place holders and omitting spaces. You must not include a space after the comma that separates the *remote-file* specification from any of the file attributes, such as *filecode*, *pext*, and so on. ssue the `FUP INFO` *local-file*, DETAIL command to obtain detailed information about the file to be copied.

  filecode
  : indicates the file code of the remote file. *filecode* is a number from 0 through 32767. When FTP is in binary transfer mode, the default file code is 0. When FTP is in ASCII transfer mode, the default file code is 101. This attribute will override the current setting of the `filecode` command.

  primary
  : indicates the primary extent size in pages (2048-byte units) of the local file. *primary* is an integer from 1 through 65535. For structured files, the default is 8 pages. For unstructured files, the default is determined internally.

    For additional information, see "Usage Notes for Primary, Secondary and Maxextents Parameters" (page 73).

secondary

indicates the secondary extent size in pages (2048-byte units) of the local file. *secondary* is an integer from 1 through 65535. For structured files, the default size is 16 pages. For unstructured files, the default is determined internally.

For additional information, see "Usage Notes for Primary, Secondary and Maxextents Parameters" (page 73).

maxextents

indicates the maximum number of extents of the remote file. *maxextents* is an integer from 1 through 978. The default value is 978 extents. When your command specifies a *maxextents* value lower than 16 but greater than 0 for a stuctured or unstructured file, FTP sets the *maxextents* value to 16. If the file you are transferring already exist on the remote system, FTP purges the remote file and creates a new file using the same name and the specified extents.

NOTE: When you are transferring a large file, you can improve performance by specifying larger extent sizes allowing the system to allocate fewer extents while the file is being copied.

For additional information, see "Usage Notes for Primary, Secondary and Maxextents Parameters" (page 73).

filetype

for structured file transfer, indicates the type of file:

e	indicates an entry-sequenced file
k	indicates a key-sequenced file
r	indicates a relative file

record-len

indicates the length of the records in a structured file. You must obtain this information using the FUP INFO command with the DETAIL option (see the Note below).

pri-key-len

indicates the primary key length in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

key-offset

indicates the key offset in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

index-blk-len

indicates the index block length in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

NOTE: To ensure that structured files are correctly transferred, you must first issue the FUP INFO command with the DETAIL option for the file you wish to transfer. All file attributes that are present in the display must be included in the put command. The following example shows the file attributes for a file named KEYSEQ:

Examples

Example 1

This example copies a local ASCII file, named `BIGEDIT`, to a remote NonStop system.

First, before starting FTP, obtain a detailed listing of the file:

```
$55> fup info $abc.source.bigedit,detail

$ABC.SOURCE.BIGEDIT                    19 May 1995,  9:16
    ENSCRIBE
    TYPE U
    CODE 101
    EXT ( 8 PAGES, 16 PAGES )
    ODDUNSTR
    MAXEXTENTS 978
    BUFFERSIZE 4096
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  10 Oct 1994,  8:01
    CREATION DATE:  10 Oct 1994,  8:00
    LAST OPEN:  19 May 1995,  9:05
    EOF: 775804 (2.4% USED)
    FILE LABEL: 250 (6.1% USED)
    EXTENTS ALLOCATED: 25
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP put command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> put $abc.source.bigedit bigedit,101,8,16,978
200 PORT command successful.
150 Opening data connection for $abc.dest.bigedit (133.255.17.1) (775804 bytes).
226 Transfer complete.
local: bigedit2 remote: $abc.dest.bigedit
746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Example 2

This example copies a local key-sequenced. structured file named `keyseq` to a remote NonStop system

First, before starting FTP, obtain a detailed listing of the file:

```
$PUBS FTPTEST 14> fup info $abc.source.keyseq,detail

$ABC.SOURCE.KEYSEQ                  22 May 1995, 11:35
    ENSCRIBE
    TYPE K
    CODE 1001
    EXT ( 6 PAGES, 6 PAGES )
    REC 4000
    BLOCK  4096
    IBLOCK 4096
    KEYLEN 4
    KEYOFF 0
    MAXEXTENTS 16
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  19 May 1995, 10:11
    CREATION DATE:  19 May 1995, 10:11
    LAST OPEN:  19 May 1995, 10:11
    EOF: 20480 (10.4% USED)
```

```
FILE LABEL: 214 (5.2% USED)
EXTENTS ALLOCATED: 2
INDEX LEVELS: 1
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP put command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> put $abc.source.keyseq keyseq,k,1001,6,6,16,4000,4,0,4096
200 PORT command successful.
150 Opening data connection for source.keyseq (133.255.17.1) (7680 bytes).
226 Transfer complete.
local: keyseq remote: source.keyseq
6000 bytes received in  0.45 seconds (13.02 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Example 3

This example copies a local ASCII file named data4a into a file named `remofile` on a remote NonStop system. The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the CODE, EXT, and MAXEXTENTS values are the same as those for BIGEDIT in Example 1.

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv6 address. Then, using the information from the FUP INFO display, issue the FTP put command:

```
\IDC15.$FITI.FTPMARCH 90> ftp  fe80::a00:8eff:fe01:7db8
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8......Established.
220 idc15.om.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.data
250 CWD command successful.
ftp> put data4a remofile,101,8,16,978
200 EPRT command successful.
150 Opening data connection for remofile
(fe80::a00:8eff:fe01:7db8,1347d).
226 Transfer complete.
Local: data4a remote: remofile
746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Example 4

This example copies a local, key-sequenced, structured file named PERFK0 to a file named keyseq on remote NonStop system.The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the TYPE, CODE, REC, IBLOCK, KEYLEN, KEYOFF, and MAXEXTENTS values are the same as those for the file in Example 2.

Next, start FTP and open a connection to the desired FTP server using its IPv6 address. Then, using the information from the FUP INFO command, issue the FTP put command:

\IDC15.$FITI.FTPMARCH 91> ftp fe80::a00:8eff:fe01:7db8 \IDC15.$FITI.FTPMARCH 91.. FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS INCORPORATED 2003 Connecting to fe80::a00:8eff:fe01:7db8.....Established. 220 idc15.wipro.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM 01MAY2003) ready. Name (fe80::a00:8eff:fe01:7db8:user): super.super 331 Password required for SUPER.SUPER. Password: 230 User SUPER.SUPER logged in. GUARDIAN API enabled ftp> cd $fiti.remote 250 CWD command successful. ftp> lcd $fiti.q9552d Local working vol.subvol: "\IDC15.$FITI.Q9552D" ftp> put $fiti.q9552d.PERFK0 keyseq, k,1001,6,6,16,4000,4,0,4096 200 EPRT command successful. 150 Opening data connection for keyseq (fe80::a00:8eff:fe01:7db8,1349d). 226 Transfer complete. local: $fiti.q9552d.PERFK0 remote: keyseq 6000 bytes received in 0.45 seconds (13.02 Kbytes/s) ftp>

The time returned on your screen represents the time used to transfer the file, not the total time used to execute the command.

Usage Notes for Primary, Secondary and Maxextents Parameters

For the PUT command, you must specify the primary extent size and secondary extent size when you specify the maximum number of extents. If you specify no preference in extent sizes, FTP chooses extents sizes on the basis of efficiency factors. These efficiency factors can change from product RVU to product RVU and might vary each time you run the program. Thus, if when you use the *maxextents* parameter to set an upper boundary on file capacity, you must also use the *primary* and *secondary* parameters to choose the appropriate extent sizes. FTP ignores the *maxextents* specification when *primary* and *secondary* specifications are absent. In such cases, FTP sets *maxextents* to its default value (978).

If you specify both primary and secondary extents without specifying *maxextents*, FTP uses the *maxextents* default value. If you do this, you might create a Format 2 file when you intended to create a Format 1 file.

If you specify the primary extent size but do not specify the secondary extent size, the secondary extent value is set to the primary extent value. If, in addition, you specify a *maxextents* value, FTP uses that value.

However, if you specify the secondary extent without specifying the primary extent size. FTP set primary extent size to its default value. If, in addition, you specify a *maxextents* value, FTP ignores it and uses the *maxextents* default value.

Use the table on the following page as a quick reference.

Primary	Secondary	Maxextents	Result
specified	specified	specified	FTP uses all the specified values.
not specified	not specified	not specified	For structured files: primary and secondary are set to the default.For unstructured files: values are determined internallyFTP sets maxextents to the default value.
not specified	not specified	specified	FTP sets primary and secondary extent size according to efficiency factors.FTP sets maxextents to the default value.

Primary	Secondary	Maxextents	Result
specified	specified	not specified	FTP set maxextents to the default value.You might create a Format 2 file when you want to create a Format 1 file.
specified	not specified	specified	FTP sets the secondary extent size to the value of primary extent size.The specified maxextents value is used.
specified	not specified	not specified	FTP sets the secondary extent size to the value of primary extent size.FTP uses the maxextents default value.

pwd

Use the pwd command to display the name of the current working directory of the remote system.

```
pwd
```

Example

To display the directory, enter the following command:

```
ftp> pwd
```

quit

Use the quit command to disconnect from the remote system and exit FTP. (The quit command performs the same operation as the bye command.)

```
quit
```

Example

To stop communicating with a remote system and exit FTP, enter the following command:

```
ftp> quit
```

quote

Use the quote command to send specific arguments verbatim to the remote FTP server. The arguments are not subjected to normal command parsing.

```
quote argument [ argument ] ...
```

argument
is a sequence of characters to be sent to the remote system without being parsed by FTP.

Examples

- The following command sends three arguments to the remote system. In this example, the debug and verbose toggles are on.

```
ftp> quote
Noop---> NOOP
200 NOOP command successful.
```

- The FTP server allows a client to access either the Guardian file system or the OSS file system. Switching back and forth between the two file systems can be accomplished using the quote command from the FTP client followed by the character string “Guardian” or “OSS”. For example:

```
ftp> quote Guardian accesses the Guardian file system.
ftp> quote OSS accesses the OSS file system.
```

Once you have logged in, you can determine the active file system using the pwd command. If the result of this command is either in the form */G/volume/subvolume* or an OSS path name, the OSS file system is active. If the result is in the form $*volume.subvolume*, the Guardian file system is active.

NOTE: The quote Guardian/OSS command is disabled for the anonymous user.

recv

Use the recv command to copy a remote file to the local system. (The `recv` command performs the same operation as the get command.)

You must have a connection established with an FTP server (called an FTP session) before you can issue the recv command. The file transfer is based on the current status of all FTP options, toggles, and macros. Use the FTP status command to display these values. If the file you wish to copy is a structured file containing embedded `<CRLF>` character sequences, you must first issue the `stru` command with the r option. See the `stru` command for more details.

The syntax for transferring ASCII or binary files is:

```
recv remote-file [ local-file[ , attribute-list ]]


where attribute-list is:


[ [filecode],[primary],[secondary],[maxextents] ]
```

The syntax for transferring structured files is:

```
recv remote-file [ local-file[ , attribute-list ]]


where attribute-list is:


[[filetype],[filecode],[primary],[secondary],[maxextents],
  [record-len],[pri-key-len],[key-offset],[index-blk-len ]]
```

remote-file
: specifies the name of the remote file to be copied. Specify *remote-file* as required by the remote system.

local-file
: specifies a name for the local file. If you omit *local-file*, FTP uses the name of the *remote file* for the new file name, after modifying the name according to the current settings of the case, ntrans, and nmap commands. If you specify a hyphen (-) as the local file, the file will be displayed on the screen only (not locally saved).

attribute-list
: is a list of file attributes for a HP NonStop file. You can specify these attributes only if the local file system is Guardian.

  You must specify the attributes in the order indicated, using commas as place holders and omitting spaces. Issue the FUP INFO *remote-file*, DETAIL command to obtain detailed information about the file to be copied.

filecode

indicates the file code of the local file. *filecode* is a number from 0 through 32767. When FTP is in binary transfer mode, the default file code is 0. When FTP is in ASCII transfer mode, the default file code is 101. This attribute will override the current setting of the filecode command.

primary

indicates the primary extent size in pages (2048-byte units) of the local file. *primary* is an integer from 1 through 65535. The default is determined internally.

secondary

indicates the secondary extent size in pages (2048-byte units) of the local file. *secondary* is an integer from 1 through 65535. For structured files, the default size is 16 pages. For unstructured files, the default is determined internally.

maxextents

indicates the maximum number of extents of the local file. *maxextents* is an integer from 1 through 978. The default value is 978 extents. When your command specifies a *maxextents* value lower than 16 but greater than 0 for a stuctured or unstructured file, FTP sets the *maxextents* value to 16.

NOTE: When you are transferring a large file, you can improve performance by specifying larger extent sizes allowing the system to allocate fewer extents while the file is being copied.

filetype

for structured file transfer, indicates the type of file:

e	indicates an entry-sequenced file
k	indicates a key-sequenced file
r	indicates a relative file

record-len

indicates the length of the records in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

pri-key-len

indicates the primary key length in a structured file. You must obtain this information using the FUP INFO command with the `DETAIL` option (see the Note below).

key-offset

indicates the key offset in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

index-blk-len

indicates the index block length in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

NOTE: To ensure that structured files are correctly transferred, you must first issue the FUP INFO command with the DETAIL option for the file you wish to transfer. All file attributes that are present in the display must be included in the recv command. The following example shows the file attributes for a file named KEYSEQ:

Examples

Example 1

This example copies a remote ASCII file, named BIGEDIT, to the local system.

First, before starting FTP, obtain a detailed listing of the file:

```
$55> fup info $ABC.source.bigedit,detail

$ABC.SOURCE.BIGEDIT                    19 May 1995,  9:16
    ENSCRIBE
    TYPE U
    CODE 101
    EXT ( 8 PAGES, 16 PAGES )
    ODDUNSTR
    MAXEXTENTS 978
    BUFFERSIZE 4096
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  10 Oct 1994,  8:01
    CREATION DATE:  10 Oct 1994,  8:00
    LAST OPEN:  19 May 1995,  9:05
    EOF: 775804 (2.4% USED)
    FILE LABEL: 250 (6.1% USED)
    EXTENTS ALLOCATED: 25
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP `recv` command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> recv $abc.source.bigedit bigedit,101,8,16,978
200 PORT command successful.
150 Opening data connection for $abc.dest.source bigedit(133.255.17.1) (775804 bytes).
226 Transfer complete.
local: bigedit remote: $abc.source.bigedit
746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

Example 2

This example copies a remote, key-sequenced. structured file named `KEYSEQ` to the local NonStop system

First, before starting FTP, obtain a detailed listing of the file:

```
$6> fup info keyseq,detail

$ABC.SOURCE.KEYSEQ                     19 May 1995,  9:17
    ENSCRIBE
    TYPE K
    CODE 1001
    EXT ( 5 PAGES, 5 PAGES )
    REC 60
    BLOCK  512
    IBLOCK 512
    KEYLEN 4
    KEYOFF 0
    MAXEXTENTS 16
    OWNER 165,86
    SECURITY (RWEP): NUNU
    DATA MODIF:  10 Oct 1994, 16:13
    CREATION DATE:  10 Oct 1994, 16:13
    LAST OPEN:  19 May 1995,  9:05
    EOF: 8704 (5.3% USED)
    FILE LABEL: 214 (5.2% USED)
```

```
    EXTENTS ALLOCATED: 1
    INDEX LEVELS: 1
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP `recv` command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> recv keyseq source.keyseq,k,1001,5,5,16,60,4,0,512
200 PORT command successful.
150 Opening data connection for source.keyseq (133.255.17.1) (7680 bytes).
226 Transfer complete.
local: keyseq remote: source.keyseq
6200 bytes sent in  0.10 seconds (60.55 Kbytes/s)
ftp>
```

Example 3

This example copies a remote ASCII file named `remofile` to a file named `data4a` on the local system. The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, you should use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the CODE, EXT, and MAXEXTENTS values are the same as those for BIGEDIT in Example 1.

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv6 address. Then, using the information from the FUP INFO display, issue the FTP recv command:

```
\IDC15.$FITI.FTPMARCH 90> ftp  fe80::a00:8eff:fe01:7db8
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8......Established.
220 idc15.om.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.data
250 CWD command successful.
ftp> recv remofile,data4a 101,8,16,978
200 EPRT command successful.
150 Opening data connection for remofile
(fe80::a00:8eff:fe01:7db8,1347d).
226 Transfer complete.
Local: data4a remote: remofile
8746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

Example 4

This example copies a remote, key-sequenced, structured file named keyseq1 to a file named keyseq2 on the local NonStop system, The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the TYPE, CODE, REC, IBLOCK, KEYLEN, KEYOFF, and MAXEXTENTS values are the same as those for the file in Example 2.

Next, start FTP and open a connection to the desired FTP server using its IPv6 address. Then, using the information from the FUP INFO command, issue the FTP recv command:

```
\IDC15.$FITI.FTPMARCH 91> ftp  fe80::a00:8eff:fe01:7db8
\IDC15.$FITI.FTPMARCH 91..
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS
INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8.........Established.
220 idc15.wipro.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.remote
250 CWD command successful.
ftp> lcd $fiti.q9552d
Local working vol.subvol:  "\IDC15.$FITI.Q9552D"
ftp> recv keyseq1 keyseq2,k,1001,6,6,4000,4,0,4096
200 EPRT command successful.
150 Opening data connection for keyseq (fe80::a00:8eff:fe01:7db8,1349d).
226 Transfer complete.
local: keyseq2 remote: keyseq1
6200 bytes sent in  0.10 seconds (60.55 Kbytes/s)
ftp>
```

remotehelp

Use the remotehelp command to request available help information from the remote FTP server.

`remotehelp [ command-name ]`

command-name

specifies the command you want described. If you omit *command-name*, a list of all commands is displayed.

The information displayed depends on the FTP server you are using.

Example

To request a list of commands, enter the following command:

```
ftp> remotehelp
```

rename

Use the rename command to change the name of a remote file.

`rename from-filename to-filename`

from-filename

specifies the current name of the file.

to-filename

specifies the new name of the file.

Specify the file names as required by the remote system.

Example

The following command changes a file name from `newfile` to backup:

```
ftp> rename newfile backup
```

The file must either be in the current working directory or you must use the remote system's scheme for indicating the location of the file as part of the name. For example, you must specify a pathname on some systems.

reset

Use the reset command to clear the reply queue in order to synchronize the command and reply sequence with the remote FTP server. Use this command when the FTP remote server violates the FTP protocol. The response to this command depends on how the remote server operates.

```
reset
```

Example

To clear the reply queue, enter the following command:

```
ftp> reset
```

rmdir

Use the rmdir command to delete a directory from a remote system.

```
rmdir directory-name
```

directory-name

specifies the directory to be deleted. Specify *directory-name* as required by the remote system.

Example

To delete a directory named memos, enter the following command:

```
ftp> rmdir memos
```

runique

Use the runique command to turn on or off a mechanism that the FTP client uses to create unique names for files copied to the local system. This mechanism is also used to process names of remote files that exceed the eight-character limit for Guardian file names.

```
runique
```

The `runique` mechanism ensures that operations complete even though a naming conflict occurs. If runique is on, FTP appends a number to a file name if the target local file name in a get or mget operation already exists. If the name contains more than six characters, it is truncated before the number is appended.

For example, assume you copy a remote file named `SALES` to the local system using a get command and do not specify a name for the local file. If the current subvolume already contains a file named `SALES`, FTP names the new file `SALES1`. If `SALES1` already exists, FTP names the new file `SALES2`. The operation will fail if 99 files (`SALES1` through `SALES99`) exist on the subvolume.

If the operation is successful, FTP displays the generated, unique name.

NOTE: FTP does not apply the runique mechanism to a file name unless the two files have the same data representation type. For example, if you transfer a binary file that has the same name as an ASCII file, FTP displays an error and does not create a new name for the file.

When you start an FTP session, runique is off.

Example

To change the setting of runique, enter the following command:

```
ftp> runique
Receive unique on.
```

send

Use the send command to copy a local file to the remote system. (The send command performs the same operation as the put command.)

You must have a connection established with an FTP server (called an FTP session) before you can issue the send command. The file transfer is based on the current status of all FTP options, toggles, and macros. Use the FTP `status` command to display these values. If the file you wish to copy is a structured file containing imbedded <CRLF> character sequences, you must first issue the `stru` command with the `r` option. See the `stru` command for more details. The syntax for transferring ASCII or binary files is:

```
send local-file [ remote-file ] [ ,attribute-list ]

where attribute-list is:

[ [filecode],[primary],[secondary],[maxextents] ]
```

The syntax for transferring structured files is:

```
send local-file [ remote-file ] [ ,attribute-list ]

where attribute-list is:

[[filetype],[filecode],[primary],[secondary],[maxextents],
  [record-len],[pri-key-len],[key-offset],[index-blk-len ]]
```

local-file
specifies the name of the local file to be copied.

remote-file
specifies the name for the remote file or remote spooler. If you specify a remote spooler, the data representation type has to be set to ASCII. If you omit *remote-file*, FTP uses the name of the *local file* for the new file name after modifying the name according to the current settings of the `case`, `ntrans` and nmap commands.

attribute-list
is a list of file attributes for a HP NonStop file. You can specify these attributes only if the remote system is a HP NonStop system.

You must specify the attributes in the order indicated, using commas as place holders and omitting spaces. Issue the `FUP INFO` *local-file*, DETAIL command to obtain detailed information about the file to be copied.

filecode
indicates the file code of the remote file. *filecode* is a number from 0 through 32767. When FTP is in binary transfer mode, the default file code is 0. When FTP is in ASCII transfer mode, the default file code is 101. This attribute will override the current setting of the `filecode` command.

primary
indicates the primary extent size in pages (2048-byte units) of the local file. *primary* is an integer from 1 through 65535. For structured files, the default is 8 pages. For unstructured files, the default is determined internally.

secondary

indicates the secondary extent size in pages (2048-byte units) of the local file. *secondary* is an integer from 1 through 65535. For structured files, the default size is 16 pages. For unstructured files, the default is determined internally.

maxextents

indicates the maximum number of extents of the remote file. *maxextents* is an integer from 1 through 978. The default value is 978 extents. When your command specifies a *maxextents* value lower than 16 but greater than 0 for a stuctured or unstructured file, FTP sets the *maxextents* value to 16. If the file you are transferring already exist on the remote system, FTP purges the remote file and recreates the file using the specified extents.

NOTE: When you are transferring a large file, you can improve performance by specifying larger extent sizes allowing the system to allocate fewer extents while the file is being copied.

filetype

for structured file transfer, indicates the type of file:

e	indicates an entry-sequenced file
k	indicates a key-sequenced file
r	indicates a relative file

record-len

indicates the length of the records in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

pri-key-len

indicates the primary key length in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

key-offset

indicates the key offset in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

index-blk-len

indicates the index block length in a structured file. You must obtain this information using the `FUP INFO` command with the `DETAIL` option (see the Note below).

NOTE: To ensure that structured files are correctly transferred, you must first issue the FUP INFO command with the DETAIL option for the file you wish to transfer. All file attributes that are present in the display must be included in the put command. The following example shows the file attributes for a file named KEYSEQ:

Examples

Example 1

This example copies a local ASCII file, named `BIGEDIT`, to a remote NonStop system.

First, before starting FTP, obtain a detailed listing of the file:

```
$55> fup info $abc.source.bigedit,detail

$ABC.SOURCE.BIGEDIT                    19 May 1995,  9:16
    ENSCRIBE
    TYPE U
    CODE 101
    EXT ( 8 PAGES, 16 PAGES )
```

```
    ODDUNSTR
    MAXEXTENTS 978
    BUFFERSIZE 4096
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  10 Oct 1994,  8:01
    CREATION DATE:  10 Oct 1994,  8:00
    LAST OPEN:  19 May 1995,  9:05
    EOF: 775804 (2.4% USED)
    FILE LABEL: 250 (6.1% USED)
    EXTENTS ALLOCATED: 25
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP send command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> send $abc.source.bigedit bigedit,101,8,16,978
200 PORT command successful.
150 Opening data connection for $abc.dest.bigedit (133.255.17.1) (775804 bytes).
226 Transfer complete.
local: bigedit2 remote: $abc.dest.bigedit
ftp>
```

Example 2

This example copies a local, key-sequenced, structured file named `keyseq` to a remote NonStop system

First, before starting FTP, obtain a detailed listing of the file:

```
$PUBS FTPTEST 14> fup info $abc.source.keyseq,detail

$ABC.SOURCE.KEYSEQ                  22 May 1995, 11:35
    ENSCRIBE
    TYPE K
    CODE 1001
    EXT ( 6 PAGES, 6 PAGES )
    REC 4000
    BLOCK  4096
    IBLOCK 4096
    KEYLEN 4
    KEYOFF 0
    MAXEXTENTS 16
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  19 May 1995, 10:11
    CREATION DATE:  19 May 1995, 10:11
    LAST OPEN:  19 May 1995, 10:11
    EOF: 20480 (10.4% USED)
    FILE LABEL: 214 (5.2% USED)
    EXTENTS ALLOCATED: 2
    INDEX LEVELS: 1
```

At the TACL prompt, start FTP and open a connection to the FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, using the information highlighted in the above example, issue the FTP send command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
```

```
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> send $abc.source.keyseq keyseq,k,1001,6,6,16,4000,4,0,4096
200 PORT command successful.
150 Opening data connection for source.keyseq (133.255.17.1) (7680 bytes).
226 Transfer complete.
local: keyseq remote: source.keyseq
6000 bytes received in  0.45 seconds (13.02 Kbytes/s)
ftp>
```

Example 3

This example copies a local ASCII file named `data4a` into a file named `remofile` on a remote NonStop system.The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the CODE, EXT, and MAXEXTENTS values are the same as those for BIGEDIT in Example 1.

At the TACL prompt, start FTP. and open a connection to the FTP server using its IPv6 address. Then, using the information from the FUP INFO display, issue the FTP `send` command:

```
\IDC15.$FITI.FTPMARCH 90> ftp  fe80::a00:8eff:fe01:7db8
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS
INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8......Established.
220 idc15.om.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.data
250 CWD command successful.
ftp> send data4a remofile,101,8,16,978
200 EPRT command successful.
150 Opening data connection for remofile (fe80::a00:8eff:fe01:7db8,1347d).
226 Transfer complete.
Local: data4a remote: remofile
746761 bytes received in  2.89 seconds (252.34 Kbytes/s)
ftp>
```

Example 4

This example copies a local, key-sequenced, structured file, named `PERFK0`, to a file named keyseq on remote NonStop system. The example assumes that both the local and remote systems support IPv6 addressing.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the TYPE, CODE, REC, IBLOCK, KEYLEN, KEYOFF, and MAXEXTENTS values are the same as those for the file in Example 2.

Next, start FTP and open a connection to the desired FTP server using its IPv6 address. Then, using the information from the FUP INFO command, issue the FTP send command:

```
\IDC15.$FITI.FTPMARCH 91> ftp  fe80::a00:8eff:fe01:7db8
\IDC15.$FITI.FTPMARCH 91..
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS
INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8.........Established.
220 idc15.wipro.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
```

```
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.remote
250 CWD command successful.
ftp> lcd $fiti.q9552d
Local working vol.subvol:  "\IDC15.$FITI.Q9552D"
ftp> send PERFK0 keyseq,k,1001,6,6,4000,4,0,4096
200 EPRT command successful.
150 Opening data connection for keyseq (fe80::a00:8eff:fe01:7db8,1349d).
226 Transfer complete.
local: $fiti.q9552d.PERFK0 remote: keyseq
6000 bytes received in  0.45 seconds (13.02 Kbytes/s)
ftp>
```

sendport

Use the `sendport` command to turn the use of the FTP server `PORT` command on or off. The `sendport` command operates as a toggle. The default setting is on.

```
sendport
```

When sendport is on, FTP attempts to use a `PORT` command when establishing a connection for each data transfer. The use of a `PORT` command can prevent delays when multiple file transfers are performed.

If you know that the FTP server is ignoring `PORT` commands and incorrectly indicating that the commands have been accepted, turn sendport off.

Example

To change the setting of sendport, enter the following command:

```
ftp> sendport
```

site

Use the site command to specify that the next command on the command line is a site-specific command.

```
site site-specific-command
```

site-specific-command

is one of the following commands when the remote system is a Guardian system: help, chmod, nocrlf, and showopen. Table 7-2 in Site-Specific Commands on page 7-4 describes the functions of these commands.

NOTE: Remote servers that do not use Guardian support a different set of site-specific commands. You can obtain a list of these commands by issuing the FTP site help command.

When you enter the site command, the FTP Client does not parse it to check the syntax. The FTP Client simply forwards it, as is, to the FTP Server. Upon receipt of the site command. the FTP Server checks the command to see whether it is supported. If it is supported, the FTP Server executes the command and returns the result to the FTP Client. If the command is not supported, the FTP Server returns a “command not understood” error message to the FTP Client.

Example

To enable the display of the open flag in the DIR command output, enter the following command:

```
ftp> site showopens on
```

status

Use the status command to display the current status of all FTP options and toggles and the names of macros currently defined.

```
status
```

Example

To display current session information, enter status at the ftp> prompt. The response is similar to this one:

```
ftp> status
Connected to warehs1.zzzco.COM.
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: (in) abc (out) cfg
Nmap: (in) $2.$1 (out) $1.$2
Hash mark printing: off; Use of PORT cmds: on
Binary filecode (used in file retrieval): 0
Local working vol.subvol: "$vol1.docs"
Macros: getf
        po
ftp>
```

stru[ct]

Use the `stru[ct]` command to specify the structure of a file to be transferred. The options are file-structure or record-structure. The file-structure option is the default.

For file-structure files, there is no internal structure; the file is a continuous sequence of data bytes.

For record-structure files, the file is made up of sequential records delimited using an embedded carriage return/line feed `<CRLF>` character sequence.

```
stru[ct] [ r | f ]
```

r

specifies a transparent record-structure file transfer

f

specifies a file-structure (or nontransparent, record-structure) file transfer. This is the default value.

Use the `stru[ct]` command with the `r` option to transparently transfer record-structure files containing embedded `<CRLF>` character sequences.

Use the `stru[ct]` command with the `f` option to return FTP to file-structure mode for transferring ASCII or binary files. Also, if the file to be transferred is a structured file that does not contain embedded `<CRLF>` character sequences, HP recommends that you transfer normally using ASCII or binary file-structure mode.

The "Structure:" field in the display for the FTP status command displays the current setting for this value. For this feature to work, both the local and remote implementations of FTP must support the struct command.

NOTE: You cannot specify alternate key parameters to create structured files.

Example 1

To transparently transfer record-structure files containing embedded `<CRLF>` character sequences, first enter the `stru r` command (to put FTP into record-structure transfer mode), then issue the `put`

or `get` command; next issue the `stru f` command (to return FTP to file-structure transfer mode). The following example illustrates this.

This example copies a local key-sequenced structured file (containing embedded <CRLF> character sequences), named `keyseq`, to a remote NonStop system.

First, before starting FTP, obtain a detailed listing of the file:

```
$PUBS FTPTEST 14> fup info $abc.source.keyseq,detail

$ABC.SOURCE.KEYSEQ                  22 May 1995, 11:35
    ENSCRIBE
    TYPE K
    CODE 1001
    EXT ( 6 PAGES, 6 PAGES )
    REC 4000
    BLOCK  4096
    IBLOCK 4096
    KEYLEN 4
    KEYOFF 0
    MAXEXTENTS 16
    OWNER 8,164
    SECURITY (RWEP): NUNU
    DATA MODIF:  19 May 1995, 10:11
    CREATION DATE:  19 May 1995, 10:11
    LAST OPEN:  19 May 1995, 10:11
    EOF: 20480 (10.4% USED)
    FILE LABEL: 214 (5.2% USED)
    EXTENTS ALLOCATED: 2
    INDEX LEVELS: 1
```

Next, start FTP and open a connection to the desired FTP server using its IPv4 address. (This example assumes that both hosts have IPv4 addresses.) Then, issue the `stru r` command. Using the information highlighted in the above example, issue the FTP put command, then issue the `stru f` command:

```
117> ftp
FTP Client - T9552D30 - (31OCT94) - COPYRIGHT TANDEM COMPUTERS INC. 1994
ftp> open 133.255.17.1
Connecting to 133.255.17.1.........Established.
220 comm.Tandem.COM FTP SERVER T9552D30 (Version 2.c TANDEM 31OCT94) ready.
Name (133.255.17.1:guest): manuals.writer
331 Password required for MANUALS.WRITER.
Password:
230 User MANUALS.WRITER logged in.
ftp> stru r
200 STRU R ok.
ftp> put $abc.source.keyseq keyseq,k,1001,6,6,4000,4,0,4096
200 PORT command successful.
150 Opening data connection for source.keyseq (130.252.12.3,4394d) (7680 bytes).
226 Transfer complete.
local: keyseq remote: source.keyseq
6000 bytes received in  0.45 seconds (13.02 Kbytes/s)
ftp> stru f
200 STRU F ok.
```

Example 2

This example is similar to Example 1. This example copies a local, key-sequenced, structured file (containing embedded <CRLF> character sequences), named `PERFK0`, to a file named `keyseq` on remote NonStop system.The example, however, assumes that both the local and remote systems use IPv6 addresses.

As in the previous examples, use a FUP INFO command to obtain a detailed listing of the file before starting FTP. In this example, assume that the TYPE, CODE, REC, IBLOCK, KEYLEN, KEYOFF, and MAXEXTENTS values are the same as those for the file in Example 1.

Next, start FTP and open a connection to the desired FTP server using its IPv6 address. Then, issue the `stru r` command. Using the information highlighted in the above example, issue the FTP `put` command, then issue the `stru f` command:

```
\IDC15.$FITI.FTPMARCH 91> ftp  fe80::a00:8eff:fe01:7db8
\IDC15.$FITI.FTPMARCH 91..
FTP Client - T9552G07 - (01MAY2003) - COPYRIGHT TANDEM COMPUTERS
INCORPORATED 2003
Connecting to fe80::a00:8eff:fe01:7db8.........Established.
220 idc15.wipro.tcpn.com FTP SERVER T9552G07 (Version 3.r TANDEM
01MAY2003) ready.
Name (fe80::a00:8eff:fe01:7db8:user): super.super
331 Password required for SUPER.SUPER.
Password:
230 User SUPER.SUPER logged in.  GUARDIAN API enabled
ftp> cd $fiti.remote
250 CWD command successful.
ftp> lcd $fiti.q9552d
Local working vol.subvol:  "\IDC15.$FITI.Q9552D"
ftp> stru r
200 STRU R ok.
ftp> put $fiti.q9552d.PERFK0  keyseq,k,1001,6,6,4000,4,0,4096
200 EPRT command successful.
150 Opening data connection for keyseq (fe80::a00:8eff:fe01:7db8,1349d).
226 Transfer complete.
local: $fiti.q9552d.PERFK0 remote: keyseq
4020000 bytes sent in 12.52 seconds (313.56 Kbytes/s)
ftp> stru f
200 STRU F ok
```

sunique

Use the sunique command to turn on or off a mechanism that the FTP server uses to create unique names for files copied to the remote system. The sunique command operates as a toggle.

```
sunique
```

The sunique mechanism ensures that operations complete even though a naming conflict occurs. If sunique is on, the FTP server uses a scheme for assigning a unique name. Each server determines its own scheme.

The FTP server must support the FTP-protocol `STOU` command in order for sunique to operate properly. (You can use the `remotehelp` command to check whether the server supports `STOU`.) If the operation is successful, the FTP server reports the unique name it creates.

When you start an FTP session, sunique is off.

Example

To change the setting of `sunique`, enter the following command:

```
ftp> sunique
Store unique on.
```

tenex

Use the `tenex` command to set the data representation type to that required to communicate with TENEX machines (a local logical byte size of 8 bits).

```
tenex
```

Example

The following command sets the data type to `tenex`:

```
ftp> tenex
```

type

Use the `type` command to set the data representation type used for data transfer and storage.

```
type   { ascii | binary | tenex }
```

`type-name`

specifies the data representation type. If you omit *`type-name`*, the default type is ascii.

See the `ascii` and `binary` command descriptions for additional information.

Examples

To change the type to binary, enter the following command:

```
ftp> type binary
ftp>
```

You can also use the `binary` command to set this type.

To reset the type to the default value, enter the following command:

```
ftp> type
Using ascii mode to transfer files.
```

user

Use the `user` command to identify yourself to the remote system.

```
user user-name [ [ password ]  account ]
```

`user-name`

is the user name or user ID by which you are known on the remote system.

`password`

is your user password for the remote system. Omit the password if you do not want it to appear on your screen. If you do not specify a password and the server requires one, FTP prompts you to provide one. FTP does not display what you type in response to the prompt.

When logging on to a NonStop host, you can specify a space for the password required by FTP if you do not have a password defined on the NonStop system.

`account`

is the name or number of your account on the remote system. If you omit *`account`* and the FTP server requires one, FTP prompts you for the account.

If you do not turn off the automatic logon when you start your FTP session, the process initiated by this command occurs automatically when you connect to a system. For more information on remote accounts, see "Logging On to a Remote System" (page 41).

Example

The following command identifies a user named erin whose password is `ejmonly` and whose account is `sales`.

```
ftp> user erin ejmonly sales
```

verbose

Use the `verbose` command to turn verbose mode on or off. This command operates as a toggle.

```
verbose
```

In verbose mode, all responses from the FTP server are displayed on your terminal, and statistics regarding the efficiency of a file transfer are reported when the transfer is complete.

When you start an FTP session, verbose is on.

NOTE: The statistics displayed by this command are approximate and should not be used to compare FTP performance with other implementations.

Example

Assume verbose is on. The following example illustrates the effect of turning verbose off:

```
ftp> put xtst123
200 PORT command successful.
150 Opening data connection for xtst123 (127.0.0.1,1198)
226 Transfer complete.
local: xtst123  remote: xtst123
45 bytes sent in 0.02 seconds (2.19 Kbytes/s)
ftp> verbose
Verbose mode off.
ftp> put ytst123
ftp>
```

When verbose is on, FTP displays information about the put operation, indicating when the connection is made, when the file transfer completes, the names of the local and remote files, and the number of bytes transferred.

?

Use the ? command to display either a list of all commands or helpful information about a specific command. (This command performs the same operation as the `help` command.)

`? [ command-name ]`

command-name
specifies an FTP command. If you omit *command-name*, a list of all FTP commands is displayed.

Example

The following command requests information about the `proxy` command:

```
ftp> ? proxy
proxy  issue command on alternate connection
```

$

Use the $ command to execute a macro.

`$macro-name [ argument [ argument ] ... ]`

macro-name
specifies one of the macros you have defined by using the `macdef` command.

argument
provides the value of an argument specified in one or more FTP commands in the macro. The first value replaces argument `$1`, the next value replaces `$2`, and so forth. If the macro is defined with the `$i` argument, each value replaces `$i` consecutively as the macro is repeatedly executed.

FTP does not expand any argument values based on wild-card characters in wild-card names before passing them to the macro.

Examples

Assume the following macro is currently defined:

```
macdef custget
ntrans ab+-/ xy
mget $i
```

To execute this macro three times using different wild-card names, enter the following command:

```
$custget a* b* c*
```

The first time the macro executes, the `mget` command uses the wild-card name `a*` to request the files. The second time it uses the wild-card name b*; the third time mget uses `c*`.

In the next example, the macro has two arguments, one for a file name and one for a directory name:

```
ftp> macdef putfile
Enter macro line by line, terminating it with a null line
cd $1
put $2
ftp> $putfile dirxyz fileabc
cd dirxyz                       FTP displays the commands put fileabc                       as they execute.
ftp>
```

7 Communicating With the FTP Server

HP NonStop TCP/IP applications include an FTP server (ftpserv) that you can connect to while using an FTP client on another network system.

FTP Server Commands

The FTP server supports the FTP commands described in Table 3 and recognizes, but does not process, any other FTP commands defined in Internet RFC 959. The FTP client can use uppercase or lowercase letters in command syntax.

If the FTP client you are using sends wild-card names to the FTP server, the server processes the wild-card characters using the scheme described with the FTP client glob command in Section 6, FTP—Transferring Files.

Table 3 FTP Commands Processed

Command	Processing Performed
ABOR	Aborts an active file transfer. The ABOR command must be preceded by a Telnet Interrupt Process (IP) signal and a Telnet Synchronization (SYNCH) signal. Syntax: `ABOR`
APPE	Appends data to a specified file on the server system. If the file does not exist, it stores the data in a new file. This command must be preceded by a PORT command.(Syntax: `APPE`*`filename`*)
CWD	Changes the current default subvolume (working subvolume) to the specified subvolume. Syntax: `CWD [ $`*`volume`*`.]subvolume`
DELE	Deletes the specified file from the server system. Syntax: `DELE`*`filename`*
EPRT	Allows specification of an extended address for the data connection. Such an address must consist of the network protocol as well as the network transport address. Syntax: `EPRT |`*`net-protocol`*`|`*`net-addr`*`|`*`tcp-port`*`|`where *`net-protocol`* is either 1 (IPv4) or 2 (IPv6), *`net-addr`* is the network address, and *`tcp-port`* is a string representation of the TCP port on which the host listens for the connection.
EPSV	Request that a server listen on a data port and wait for a connection. It takes an optional argument. When you issue an EPSV with no argument, the server chooses the protocol for the data connection on the basis of the protocol used for the control connection. Syntax: `EPSV`To request the use of a specific protocol. Syntax: `EPSV`*`net-protocol`*where *`net-protocol`* is either `1` (IPv4) or `2` (IPv6).
HELP	Provides a list of the FTP commands that the server can process, or describes the specified command. Syntax: `HELP [ `*`command`*` ]`
LIST	Gives detailed information on files on a specified subvolume, or information on a specified file. If the subvolume is not specified, the information is about the current default subvolume. This command must be preceded by a PORT command. Syntax: `LIST [ `*`subvolume`*`.* | filename ]`
MKD	Creates a directory using the specified *`directory-name`*. Syntax: `MKD`*`directory-name`*)

Table 3 FTP Commands Processed *(continued)*

Command	Processing Performed
MODE	Sets the data transmission mode. Only stream mode is supported. Syntax: `MODES`
NLST	Provides the names of files on a specified subvolume of the server system or on the default subvolume, if none is specified. Syntax: `NLST [ subvolume.* | filename ]`
NOOP	Initiates only an OK reply from the server. Syntax: `NOOP`)
PASS	Specifies the user's password. This command must be preceded by the USER command. If you have no password on a NonStop system, use a space to specify the password required by FTP. Syntax: `PASSpassword`
PASV	Prepares for third-party (server-to-server) transfers. When issued by a client on host A, this command requests the server on host C to wait for a connection and listen on a data port that is not its default data port. The host C server sends the host address and port that it is listening on to host A. See "Third-Party File Transfers" in this section for an example. Syntax: `PASV`)
PORT	Specifies the address of a data-connection port; this port is normally not required because the default server and client ports are used. The FTP client must specify the address as a concatenation of the 32-bit internet host address and the 16-bit port address. The address must be divided into 8-bit fields (`h1` through `h4` and `p1` and `p2`). Each field must be transmitted as a decimal number (in ASCII representation). The fields must be separated by commas. Syntax: `PORTh1, h2, h3, h4, p1, p2`
PWD	Sends the name of the current subvolume to the client. Syntax: `PWD`
QUIT	Terminates an FTP connection between the client and server. Syntax: `QUIT`
RETR	Copies a specified file from the server system to the client system. This command must be preceded by the PORT command. Syntax: `RETRfilename`
RMD	Removes the specified directory. Syntax: `RMDdirectory-name`
RNFR	Specifies the current name of a file to be renamed on the server system. This command should be followed immediately by the RNTO command. Syntax: `RNFRfilename`
RNTO	Assigns a new name to the file identified in the immediately preceding RNFR command. Syntax: `RNTOfilename`
SITE	Precedes site-specific commands on the command line. Site-specific commands are described in Table 4.
STOR	Accepts data and stores it in a file on the server system. If the file exists, the contents are replaced; otherwise, a new file is created. This command must be preceded by the PORT command. Syntax: `STORfilename`.When storing a file on a Guardian system with a binary transmission mode, you can also specify the file code, the primary and secondary extent sizes, and the maximum number of extents. See the put command in Section 6, FTP—Transferring Files, for the required syntax.

Table 3 FTP Commands Processed *(continued)*

Command	Processing Performed
STOU	Performs a STOR operation but assigns a unique file name. See the description of the FTP runique command in Section 6, FTP—Transferring Files, for details about the naming scheme used. Syntax: `STOU`*`filename`*
STRU	Sets the transfer structure to record-structure file or file-structure file. Syntax: `STRU { r | f }`
TYPE	Sets the data representation type, which can be A for ASCII text files or B for binary data. Syntax: `TYPE { A | B }`
USER	Specifies your user ID or user name for logging on to the server system. A password is required; see the PASS command. Syntax: `USER`*`username`*
XCWD	Performs the same function as the CWD command. Syntax: `XCWD [$`*`volume`*`.]`*`subvolume`*
XMKD	Performs the same function as the MKD command. Syntax: `XMKD`*`directory-name`*
XPWD	Performs the same function as the PWD command. Syntax: `XPWD`

Site-Specific Commands

Site-specific commands specify services provided by the FTP Server that are essential to file transfer but are not included as commands in the protocol. On the command line, the SITE command and the subsequent site-specific commands are case-insensitive.

Table 4 SITE Commands

Command	Processing Performed
HELP	Provides a functional description of a specified site-specific command or If no command name is specified, provides a complete list of site-specific commands.(Syntax: `SITE HELP [ `*`command-name`*` ]`
CHMOD	Sets the security string for a file to be transferred to the OSS file system. The security string must be a three digit octal value. The default value is the octal value 666.(Syntax: `SITE CHMOD [ `*`security-string`*` ]`An invalid security string specified in the SITE CHMOD command causes the return of an error message to the Client. The previous, and valid, value is retained.
NOCRLF	Enables or disables the appending of carriage-return line feeds to records during structured file transfers.(Syntax: `NOCRLF [ on | off ]`)
SHOWOPEN	Enables or disables the display of the open flag in the DIR command output. The open flag is the “o” placed between the file name and file code to indicate either that the file is open or that it has an incomplete TMF transaction against it. The FTP Server’s default behavior is to suppress the open flag in the DIR command output.(Syntax: `SITESHOWOPEN [ on | off ]`)

Third-Party File Transfers

A third-party file transfer is an operation in which you establish a primary connection to an FTP server on host B (from your client on host A) and then establish a secondary connection to an FTP server on host C. After both connections are established, you can issue a command on host A to

transfer files from host B to host C by the third party (host A). The NonStop FTP server can perform the role of host B (or host C) if the other server in the operation supports the PASV command.

The following example illustrates, in a general way, how issuing FTP proxy commands at host A can establish connections and transfer files. Your FTP client may, however, issue a different ftp prompt from the one shown (ftp>).

Host B is named phost (primary host) and has internet address 50.0.0.3. Host C is named shost (secondary host) and has internet address 50.0.0.1. The FTP commands that the server receives from the client are displayed following the symbol --->. (Normally, these commands are displayed only when the FTP client you are using is in debug mode.)

The comments in the right column indicate the action requested by each FTP client command and the destination of FTP commands issued by the FTP client.

```
TACL 4> ftp phost                  Connect to primary host.       .                              User then logs
 on to host (not        .                              shown, depends on system).  ftp> proxy open shost
        Connect to secondary host.       .                              User then logs on to       .
                  secondary host (not shown).       .
ftp> proxy get pfile sfile         Copy pfile from primary       .                              host to sfile
on secondary       .                        host.
---> PASV                          Issued to secondary host.
shost:227 Entering Passive Mode (50,0,0,3,4,55).

---> PORT 50,0,0,3,4,55            Issued to primary host.
phost:200 PORT command okay.

---> RETR pfile                    Issued to primary host.
phost:150 Opening data connection for pfile
(50.0.0.3.1079)
(30 bytes).

---> STOR sfile                    Issued to secondary host.
shost:150 Opening data connection for sfile(50.0.0.1.20)
shost:226 Transfer complete.
phost:226 Transfer complete.
```

Users of NonStop TCP/IPv6 can specify IPv6 addresses to establish connections for third-party transfer operations. Note that all hosts involved in such transactions must have IPv6 addresses. Thus, when an FTP client uses an IPv6 address to connect to the primary host, the FTP client must use an IPv6 address to connect to the secondary host (proxy server).

In the following example, the primary host has an Internet address of fe80::a00:8eff:fe01:7db8, and the secondary host has an address of fe80::a00:8eff:fe00:8e71. Note that the EPSV and EPRT commands are issued instead of PASV and PORT.

```
TACL 4> ftp fe80::a00:8eff:fe01:7db8       Connect to primary host.       .                               User
then logs on to host (not        .                              shown, depends on system).  ftp> proxy open
fe80::a00:8eff:fe00:8e71   Connect to secondary host.       .                              User then logs on to
      .                              secondary host (not shown).
ftp> proxy get pfile sfile               Copy pfile from primary       .                               host to
 sfile on secondary       .                        host.
---> EPSV 2                              Issued to secondary host

fe80::a00:8eff:fe00:8e71:229 Entering Extended Passive Mode (|||1342|)

---> EPRT |2|fe80::a00:8eff:fe00:8e71|1342|   Issued to primary host.
fe80::a00:8eff:fe01:7db8:200 EPRT command successful.

---> RETR pfile                             Issued to primary host fe80::a00:8eff:fe01:7db8:150 Opening data
 connection for pfile
(fe80::a00:8eff:fe00:8e71,1342d) (30 bytes).


---> STOR sfile                             Issued to secondary host. fe80::a00:8eff:fe00:8e71:150 Opening
data connection for sfile (fe80::a00:8eff:fe01:7db8,5342d). fe80::a00:8eff:fe00:8e71:226 Transfer complete.
fe80::a00:8eff:fe01:7db8:226 Transfer complete.
```

For information about proxy processing from a NonStop FTP client, see the proxy command in Section 6, FTP—Transferring Files.

Transferring Structured Files

The following is some information about structured-file transfers. The syntax for the commands used to transfer structured files is given first. The syntax is followed by an example program. Refer to the `append`, `get`, `put`, `recv`, and `send` command descriptions in Section 6, FTP—Transferring Files, for detailed information about these commands.

Command Syntax

The syntax for the commands used to transfer structured files is:

```
append local-file [ remote-file ]

get remote_file [ local-file  [ , attribute-list ] ]

put local-file [ remote-file [ , attribute-list ] ]

recv remote_file [ local-file  [ , attribute-list ] ]

send local-file [ remote-file [ , attribute-list ] ]
```

The *attribute-list* is:

```
[   [ filetype     ]
  , [ filecode     ]
  , [ primary      ]
  , [ secondary    ]
  , [ maxextents   ]
  , [ record-len   ]
  , [ pri-key-len  ]
  , [ key-offset   ]
  , [ index-blk-len ] ]
```

filetype
e (entry-Sequenced file) k (key-Sequenced file) r (relative file)

filecode
file code of the file to be created

primary
primary extent size

secondary
secondary extent size

maxextents
maximum extent size

record-len
maximum length of the record in bytes

pri-key-len
primary key length

key-offset
key offset

index-blk-len
length of the index block

Example

For this example assume the following:

abc	is an ASCII-coded file
ebig	is an entry-sequenced file with a file code of 1001
kbig	is a key-sequenced file with a file code of 1002
rbig	is a relative file with a file code of 1003

The program:

```
STRU F
put abc ebig using the attributes for ebig file,
 overwrite its content with abc.
 ebig is still an entry-sequenced file.
put abc kbig overwrites kbig with abc.
put abc rbig overwrites rbig with abc.
put ebig ebig1,e,1002,,,,60 creates ebig1 with attributes specified.
get kbig kbig1,k,1002,,,,500,20 creates kbig1 with attributes specified.
append ebig rbig append records from an entry-sequenced
   file to a relative file.
put kbig abig create an ascii file, abig, with information
   from a key-sequenced file, kbig.
get rbig rbig1,r,1003,,,,60 creates a relative file, rbig1.

STRU R
put ebig ebig2 creates an entry-sequenced file with a
   default record length of 4000.
append ebig ebig2 append record in ebig to the end of ebig2
get kbig ebig3 creates an entry-sequenced file, ebig3, using
   the data from kbig.
put ebig kbig2,k,e,1001,,,,60 creates a key-sequenced file, kbig2.
append rbig rbig2,r,1003,,,,112 creates a relative file, rbig2.
get ebig ebig4,e,1001,1000,,,60 creates an entry-sequenced file, ebig4.
```

Implementation Considerations

With some sites being naturally file-oriented and others naturally record-oriented there can be problems when a file with one structure is sent to a host oriented to the other structure. For example, sending an HP NonStop structured file to a Unix machine, which treats all files as a byte stream.

The stru command provides transparent data transfers. By default, file-structure is assumed for data transfers. Use the stru command with the r option specified to transfer record-structure files containing embedded <CRLF> character sequences.

In the HP NonStop implementation, a <CRLF> encountered in a file is treated as a record delimiter for a structured-file transfer. If a <CRLF> appears within a record, it is misinterpreted as marking an end-of-record. Using the `stru` command with the `r` option specified `(stru r)` provides transparent data transfer.

Transferring files using `stru r` (record-structure mode) incurs extra processing. Therefore, the throughput can be significantly lower (roughly 50%).

No other vendor has implemented record-structure file transfer. Therefore, its usefulness is limited to the NonStop host that has implemented this feature.

If you accidentally turn on record-structure mode, and the other side hasn't implemented this feature, your original data will be mixed with Telnet control code.

Due to limitations of the underlying library routines we are using, records in a key-sequenced file must be in sorted order before you use the `put` or `send` commands.

Disallowing Logons

The system administrator can disallow logons by simply creating a file, named FTPUSERS, secured for restricted access. FTPUSERS is an ASCII-coded edit file (type 101) that resides on $SYSTEM.ZTCPIP.

The FTPUSERS file contains a list of users for whom logons are disallowed. Comment lines can be prefixed with the number symbol (#) in the first column. User names and aliases are entered into this file by the system administrator. The user names and aliases that do not conform to the Guardian *group-name*,*user-name* format are case sensitive when checked by the FTP server to determine which logons are disallowed. The user names in the Guardian format are not case sensitive when checked.

NOTE: Aliases associated with the OSS anonymous user are a special case and are required only to be entered in the FTPUSERS file as lowercase characters.

Example

The following example illustrates an FTPUSERS file:

```
# This file describes the names of the users for which
# FTP logons will be denied. The name of the file is
# $SYSTEM.ZTCPIP.FTPUSERS

super.super
super.operator
null.ftp
guest
student
OssGuest
ftp
```

Accessing Either Guardian or OSS File Systems

The FTP server allows a client to access either the Guardian file system or the OSS file system. Switching back and forth between the two file systems can be accomplished using the quote command from the FTP client followed by the character string “Guardian” or “OSS”.

For example:

```
quote Guardian accesses the Guardian file system.
quote OSS accesses the OSS file system.
```

Once you have logged in, you can determine the active file system using the pwd command. If the result of this command is either in the form */G/volume/subvolume* or an OSS path name, the OSS file system is active. If the result is in the form $*volume.subvolume*, the Guardian file system is active.

NOTE: The quote Guardian/OSS command is disabled for the anonymous user.

8 FTP API External Specification

The FTP API external specification describes the external interface offered by the FTP application program interface (API).

Key Objectives and Features

The API is designed to allow application programs to access the HP NonStop File Transfer Protocol (FTP) programmatically.

The API is a set of library routines. The application program links its object code with the API library object file to create an executable image file.

Any changes in internal design of the API should have no impact on existing applications using the API. The API supports nowait, and indefinite wait operations.

There can be a maximum of one I/O operation on each connection. For the nowait operation, you must verify this. As with other NSK procedure calls, when using the nowait option, do not tamper with the buffer before the I/O is completed.

Error handling is standardized. A return code of -1 signifies that the command has failed, and the error number gives the reason. (See Appendix B for error definitions).

The API offers debugging routines, which will display the commands and responses it sent and received, and print these messages out to the log file specified by the user.

If the program abends, or otherwise contains an error, the program terminates. All Processes started by the application should be cleared. That is, there should not be any orphan processes consuming system resources.

△ **CAUTION:** When compiling the C or C++ application, always specify wide pragmas.

Related Documents

- RFC 959
- RFC 1123
- *TCP/IP and TCP/IPv6 Programming Manual*

Connection Management

The FTP commands provide the functions described in Table 5.

Table 5 FTP Command Functions

Command	Function
`FTPoption`	Allows user to provide FTP runtime options.
`FTPclose`	Disconnects from remote FTP server, but does not exit from FTP.
`FTPopen`	Establishes a connection to the remote host. (wait)
`FTPopen_nw`	Establishes a connection to the remote host. (nowait)
`FTPproxy`	Establishes a proxy connection to the remote host. (wait)
`FTPproxy_nw`	Establishes a proxy connection to the remote host. (nowait)

Table 5 FTP Command Functions *(continued)*

Command	Function
`FTPconnect`	Allows user to connect to another host after logging off from the remote host using FTPclose. (wait)
`FTPconnect_nw`	Allows user to connect to another host after logging off from the remote host using FTPclose. (nowait)
`FTPcontimer`	Sets the value for the length of time the client waits for an incoming connection request from the server. (wait)
`FTPcontimer_nw`	Sets the value for the length of time the client waits for an incoming connection request from the server. (nowait)
`FTPlogin`	Allows user to log on as another user for the existing connection. (wait)
`FTPlogin_nw`	Allows user to log on as another user for the existing connection. (nowait)
`FTPbye`	Disconnects from the remote FTP server and exits FTP.

File Management

The FTP commands that support file management are depicted in Table 6.

Table 6 File-Management Commands

Command	Function
`FTPappend`	Appends a local file to a remote file. (wait)
`FTPappend_nw`	Appends a local file to a remote file. (nowait)
`FTPasline`	Sets the aslinemode option. (wait)
`FTPasline_nw`	Sets the aslinemode option. (nowait)
`FTPcd`	Changes the directory on the remote host. (wait)
`FTPcd_nw`	Changes the directory on the remote host. (nowait)
`FTPdelete`	Deletes a file from the remote host. (wait)
`FTPdelete_nw`	Deletes a file from the remote host. (nowait)
`FTPget`	Retrieves a file from the remote host. (wait)
`FTPget_nw`	Retrieves a file from the remote host. (nowait)
`FTPlcd`	Changes the directory on a local machine. (wait)
`FTPlcd_nw`	Changes the directory on a local machine. (nowait)

Table 6 File-Management Commands *(continued)*

Command	Function
`FTPmkdir`	Creates a directory on remote host. (wait)
`FTPmkdir_nw`	Creates a directory on remote host. (nowait)
`FTPput`	Stores a file onto a remote host. (wait)
`FTPput_nw`	Stores a file onto a remote host. (nowait)
`FTPquote`	Sends specific arguments verbatim to the remote host. The arguments are not subjected to normal command parsing. (wait)
`FTPquote_nw`	Sends specific arguments verbatim to the remote host. The arguments are not subjected to normal command parsing, (nowait)
`FTPrename`	Renames a file on the remote host. (wait)
`FTPrename_nw`	Renames a file on the remote host. (nowait)
`FTPreset`	Clears the reply queue in order to maintain synchronization with remote FTP server. (wait)
`FTPreset_nw`	Clears the reply queue in order to maintain synchronization with remote FTP server. (nowait)
`FTPrmdir`	Removes a directory on the remote host. (wait)
`FTPrmdir_nw`	Removes a directory on the remote host. (nowait)
`FTPsetfilecode`	Sets the file code. (wait)
`FTPsetfilecode_nw`	Sets the file code. (nowait)
`FTPsetstruct`	Sets the file structure to either page, file or record. (wait)
`FTPsetstruct_nw`	Sets the file structure to either page, file or record. (nowait)
`FTPsettype`	Sets the transfer type to either ascii, binary or tenex mode. (wait)
`FTPsettype_nw`	Sets the transfer type to either ascii, binary or tenex mode. (nowait)

Miscellaneous Functions

The FTP commands in Table 7 provide additional status information.

Table 7 FTP Miscellaneous Functions

Command	Function
FTPerrmsg	Prints the error message text associated with the current error number to the stdout.
FTPerrtext	Returns the error text associated with the current error number to the application making the call.
FTPio	Checks that I/O status is completed.
FTPlist	Returns a list of file names that matches the pattern specified. (wait)
FTPlist_nw	Returns a list of file names that matches the pattern specified. (nowait)
FTPpwd	Returns the name of the current working directory. (wait)
FTPpwd_nw	Returns the name of the current working directory. (nowait)
FTPreplytext	Retrieve the reply text to buffer up to the buffer size-limit. (wait)
FTPreplytext_nw	Retrieve the reply text to buffer up to the buffer size-limit.(nowait)

API Routines

The following sections describe the API routines.

Append Routine

The routines FTPappend and FTPappend_nw append a local file to a remote one.

```
#include "ftpexth"
short FTPappend(short hd, short proxy, char *in,
                char *out)
short FTPappend_nw(short hd, short proxy, char *in,
                char *out, long tag)
```

hd

File number.

proxy

1 , this command applies to the proxy connection.

0 , this command applies to the primary connection.

in

local file name to be appended to the remote file.

out

remote file name.

tag

tag parameter to be used for GUARDIAN nowait operation initiated by FTPappend_nw.

The return value is -1 if it fails, otherwise it is successful.

Line Mode Option

The routines FTPasline and FTPasline_nw set the aslinemode option.

NOTE: FTPasline and FTPasline_nw function only for a direct connection. You cannot use them with proxy connections.

```
#include "ftpexth"
short FTPasline(short hd, short proxy, short mode)
short FTPasline_nw(short hd, short proxy, short mode,
                   long tag)
```

hd

File number.

proxy

1 , his command applies to the proxy connection.Do not specify 1. If you do, you will receive an error 414 (NOT_PROXY) when the call executes.

0 , this command applies to the primary connection.

mode

valid options includes: NONE, WRAP and CUT.

tag

tag parameter to be used for GUARDIAN nowait operation initiated by FTPappend_nw.

The return value is -1 if it fails, otherwise it is successful.

Close Connection

The routine FTPbye closes the connection, and deallocates resources associated with the connection.

```
#include "ftpexth"
short FTPbye(short hd, short proxy)
```

hd

File number.

proxy

1 , this command applies to the proxy connection.

0 , this command applies to the primary connection.

The return value is -1 if it fails, otherwise it is successful.

Change Directory—Remote Machine

The routines FTPcd and FTPcd_nw change the directory on a remote machine.

```
#include "ftpexth"
short FTPcd(short hd, short proxy, char *dir)
short FTPcd_nw(short hd, short proxy, char *dir,
               long tag)
```

hd

File number.

proxy

1 , this command applies to the proxy connection.

0 , this command applies to the primary connection.

dir

target directory name.

tag

parameter to be used for the GUARDIAN nowait operation initiated by FTPcd_nw.

The return value is -1 if the routine fails, otherwise the routine is successful. The error number is set accordingly.

Disconnect From Remote Server

The routine FTPclose disconnects the user from a remote FTP server, but does not exit from FTP. Thus, the user can connect to another host without invoking FTP again.

```
#include "ftpexth"
short FTPclose(short hd)
```

hd
: File number.

The return value is -1 if the routine fails, otherwise the routine is successful. The error number is set accordingly.

Connect to Host

The routines FTPconnect and FTPconnect_nw allow the user to connect to another host after logging off from the remote host, using FTPclose.

```
#include "ftpexth"
short FTPconnect(short hd, char *host,
                 short port, char *user,
                 char *passwd, char *acct)
short FTPconnect_nw(short hd, char *host,
                    short port, char *user,
                    char *passwd, char *acct,
                    long tag)
```

hd
: File number.

host
: name of the remote host.

port
: port number to be used. If the port number specified <0, the system will choose its own port number.

user
: user name used to log on.

passwd
: password.

acct
: account to be used to log on. If acct is set to "", it is considered unspecified.

tag
: parameter to be used for the GUARDIAN nowait operation initiated by FTPconnect_nw.

The return value is -1 if the routine fails, otherwise the routine is successful. The error number is set accordingly.

Delete File on Remote Computer

The routines FTPdelete and FTPdelete_nw delete a file on a remote machine.

```
#include "ftpexth"
short FTPdelete(short hd, short proxy,
                char *filename)
short FTPdelete_nw(short hd, short proxy,
                   char *filename, long tag)
```

hd
File number.

proxy
1 , this command applies to the proxy connection.

0 , this command applies to the primary connection.

filename
Name of the file to be deleted.

tag
parameter to be used for the GUARDIAN nowait operation initiated by FTPdelete_nw.

The return value is -1 if the routine fails, otherwise the routine is successful. The error number is set accordingly.

Error Text

The routine FTPerrtext returns the error text associated with the current errno up to the size limit of the buffer.

```
#include "ftpexth"
void FTPerrtext(char *buffer, short limit);
```

buffer
pointer to where the data is returned.

limit
maximum number of bytes that the buffer can hold. If the return list is longer than the limit, the error number will be set to TRUNCATED and data up to `limit size` will be returned in the buffer.

Print Error Message

The routine FTPerrmsg prints the error message text associated with the current error number to standard output.

```
#include "ftpexth"
void FTPerrmsg(char *s)
```

s
text printed preceding the error message.

Return value
none.

Retrieve File From Remote Host

The routines FTPget and FTPget_nw retrieve a file from a remote host.

```
#include "ftpexth"
short FTPget(short hd, short proxy, char *in,char *out)
short FTPget_nw(short hd, short proxy, char *in,
                char *out, long tag)
```

hd
file number.

proxy
1, this command applies to the proxy connection.

0, this command applies to the primary connection.

in

remote file name.

out

a string containing the local file name and a list of attributes for the local file. You must specify the attributes in the order indicated below; use commas as placeholders and omit spaces.

The syntax for transferring ASCII or binary files is:

```
[[ filecode],[ primary],[ secondary],[ maxextents]]
```

Thesyntax for transferring structured files is:

```
[[ filetype],[ filecode],[ primary],[ secondary],
API Routines 115
[ maxextents],[ record-len],[ pri-key-len],
[ key-offset],[ index-blk-len ]]
```

If "out" is left unspecified, by setting out as"", the local file name will be the same as the remote file name (that is, in) and the file attributes take the default values.

filetype

indicates the type of file for structured file transfer:

e: indicates an entry-sequenced file.

k: indicates a key-sequenced file.

r: indicates the relative file.

filecode

indicates the file code of the local file. File code is a number between 0 and 32767. When FTP is in primary transfer mode (set by calling FTPsettype () and specifying type `I`), the default code is zero (0). When FTP is in ASCII mode (set by calling FTPsettype () and specifying type `A`), the default code is 101. The file code attribute overrides the current setting of FTPsetfilecode ().

primary

indicates the primary extent size in pages (2048-byte units) of the local file. primary is an integer from 1 through 65535. The default is determined internally.

secondary

indicates the secondary extent size in pages (2048-byte units) of the local file. secondary is an integer from 1 through 65535. For structured files, the default size is 16 pages. For unstructured files, the default is determined internally.

maxextents

indicates the maximum number of extents of the local file. maxextents is an integer from 1 through 978. The default value is 978 extents.

record-len

indicates the length of the records in a structured file.

pri-key-len

indicates the primary key length in a structured file.

key-offset

indicates the key offset in a structured file.

index-blk-len

indicates the index block length in a structured file.

tag

parameter to be used for the GUARDIAN nowait operation initiated by FTPget_nw.

The return value is -1 if the routine fails, otherwise the routine is successful. The error number is set accordingly.

Check I/O Status

The routine FTPio checks whether the I/O associated with the connection has been completed.

```
#include "ftpexth"
short FTPio(short *hd, long wait, char **buffer,
            long *tag)
```

hd
: file number.

wait
: <>0,wait indefinitely for this command to.

  0 , returns to the caller immediately, regardless of whether or not an I/O completion occurs

buffer
: pointer to where the data is returned.

tag
: same as tag parameter in AWAITIOX.

The return value is 1 if I/O is completed, 0 if I/O is not completed yet, -1 if there is an error.

Change Directory on Local Machine

The routines FTPlcd and FTPlcd_nw change the directory on a local machine.

NOTE: FTPlcd and FTPlcd_nw function only when there is a direct connection. You cannot use them with proxy connections.

```
#include "ftpexth"
short FTPlcd(short hd, short proxy, char *ldir)
short FTPlcd_nw(short hd, short proxy, char *ldir,
                long tag)
```

hd
: file number.

proxy
: 1 , this command applies to the proxy connection. Do not specify 1. If you do, you will receive an error 414 (NOT_PROXY) when the call executes.

  0 , this command applies to the primary connection.

ldir
: local target directory.

tag
: parameter to be used for the GUARDIAN no wait operation initiated by FTPlcd_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Display List of Files

The routines FTPlist and FTPlist_nw return a list of the files matching the specified pattern.

```
#include "ftpexth"
short FTPlist(short hd, short proxy, char *pattern,
              char *buffer, short limit)
short FTPlist_nw(short hd, short proxy,char *pattern,
                 char *buffer, short limit, long tag)
```

hd
: file number.

proxy
1, this command applies to the proxy connection.
0, this command applies to the primary connection.

pattern
pattern to be matched.

buffer
buffer area to hold the return results. The list of file names will be separated by CRLF.

limit
maximum number of bytes that the buffer can hold. If the return list is longer than the limit, the error number will be set to TRUNCATED, and data up to limit size will be returned in the buffer.

tag
parameter to be used for the GUARDIAN nowait operation initiated by FTPlist_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Log On as Another User

The routines FTPlogin and FTPlogin_nw allow the user to log on as another user on an existing connection.

```
#include "ftpexth"
short FTPlogin(short hd, short proxy,char *user,
               char *passwd, char *acct)
short FTPlogin_nw(short hd, short proxy, char *user,
                  char *passwd, char *acct, long tag)
```

hd
File number.

proxy
1, this command applies to the proxy connection.
0, this command applies to the primary connection.

user
username used to log on.

passwd
password.

acct
account to be used to log on. If acct is set to "", then it is considered unspecified.

tag
parameter to be used for the Guardian nowait operation initiated by FTPlogin_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Create Directory on Remote Machine

The routines FTPmkdir and FTPmkdir_nw create a directory on the remote machine.

```
#include "ftpexth"
short FTPmkdir(short hd, short proxy, char *dir)
short FTPmkdir_nw(short hd, short proxy, char *dir,
                  long tag)
```

hd
file number.

proxy

1, this command applies to the proxy connection.

0, this command applies to the primary connection.

dir

name of the directory to be created.

tag

parameters to be used for Guardian nowait operation initiated by FTPmkdir_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Open Connection to Remote Host

The routines FTPopen and FTPopen_nw open a connection to the remote host.

```
#include "ftpexth"
short FTPopen(char *host, short port, char *user,
              char *passwd, char *acct, char *log)
short FTPopen_nw(char *host, short port, char *user,
                 char *passwd, char *acct, char *log,
                 long tag)
```

host

name of the remote host.

port

port number to be used. If the port number specified < 0, then the system will choose its own port number.

user

username used to logon.

passwd

password.

acct

account to be used to log on. If acct is set to "", then it is considered as unspecified.

log

name of the log file to be used. All the commands sent and responses received are logged to the log file. If log file is specified as:

"" - all log messages will be redirected to the home terminal.

$null - all log messages will be discarded.

tag

parameters to be used for the Guardian nowait operation initiated by FTPopen_nw.

The return value is -1 if the routine fails. The error number is set accordingly. Otherwise, it is the file number associated with this connection.

Establish Proxy Connection to Remote Host

The routines FTPproxy and FTPproxy_nw establishes a proxy connection to the remote host.

```
#include "ftpexth"
short FTPproxy(short hd, char *host, short port,
               char *user, char *passwd, char *acct)
short FTPproxy_nw(short hd, char *host, short port,
                  char *user, char *passwd, char *acct,
                  long tag)
```

hd
: file number.

host
: name of the remote host.

port
: port number to be used. If the port number specified < 0, the system will choose its own port number.

user
: username used to logon.

passwd
: password.

acct
: account to be used to log on. If `acct` is set to "", it is considered unspecified.

tag
: parameter to be used for the Guardian nowait operation initiated by FTPproxy_nw.

The return value is -1 if the routine fails and the error number is set accordingly. Otherwise the routine is successful.

Store a File onto a Remote Host

The routines FTPput and FTPput_nw store a file onto a remote host.

```
#include "ftpexth"
short FTPput(short hd, short proxy, char *in,char *out)
short FTPput_nw(short hd, short proxy, char *in,
                char *out, long tag)
```

hd
: file number.

proxy
: 1, this command applies to the proxy connection.

  0, this command applies to the primary connection.

in
: local file name. The local file must be an Enscribe disk file. It can't be a temporary file, a running process, or a Spooler location.

out
: a string containing the remote file name and a list of attributes for the remote file. You must specify the attributes in the order indicated below, use commas as placeholders, and omit spaces.

  if "out" is left unspecified, by setting out as "", the remote file name will be the same as the local file name (that is, in) and the file attributes take the default values.

  filecode
  : indicates the file code of the remote file. file code is a number between 0 and 32767. When FTP is in binary transfer mode (set by calling FTPsettype() and specifying type I), the default file code is 0.When FTP is in ASCII mode (set by calling FTPsettype() and specifying type A, the default file code is 101. The filecode attribute overrides the current setting of FTPsetfilecode().

primary
indicates the primary extent size in pages (2048-byte units) of the remote file. primary is an integer from 1 through 65535. For structured files, the default size is 8 pages. For unstructured files, the default is determined internally.

secondary
indicates the secondary extent size in pages (2048-byte units) of the remote file. secondary is an integer from 1 through 65535. For structured files, the default size is 16 pages. For unstructured files, the default is determined internally. maxextents indicates the maximum

maxextents
indicates the maximum number of extents of the remote file. maxextents is an integer from 1 through 978. The default value is 978 extents.

filetype
for structured file transfer, indicates the type of file:

e: indicates an entry-sequenced file.

k: indicates a key-sequenced file.

r: indicates a relative file.

record-len
indicates the length of the records in a structured file.

pri-key-len
indicates the primary key length in a structured file.

key-offset
indicates the key offset in a structured file.

index-blk-len
indicates the index block length in a structured file.

tag
parameter to be used for the Guardian nowait operation initiated by FTPput_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

For additional information on specifying primary, secondary, and maxextents values, see "Retrieve File From Remote Host" (page 105).

Display Current Working Directory

The routines FTPpwd and FTPpwd_nw returns the name of the current working directory.

```
#include "ftpexth"
short FTPpwd(short hd, short proxy, char *buffer,
             short limit)
short FTPpwd_nw(short hd, short proxy, char *buffer,
                short limit, long tag)
```

hd
file number.

proxy
1, this command applies to the proxy connection.

0, this command applies to the primary connection.

buffer
data buffer to hold the return data.

limit
maximum number of bytes the buffer can hold. If returned data size exceeds the buffer size, then error number will be set to TRUNCATED, and the return data is set to the size of limit.

tag
parameter to be used for the GUARDIAN nowait operation initiated by FTPpwd_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Send Arguments to Remote Host

The routines FTPquote and FTPquote_nw send specific arguments verbatim to the remote host.

```
#include "ftpexth"
short FTPquote(short hd, short proxy, char *cmd)
short FTPquote_nw(short hd, short proxy, char *cmd,
                                long tag)
```

hd
file number.

proxy
1, this command applies to the proxy connection.

0, this command applies to the primary connection.

cmd
arguments to be sent to remote host.

tag
parameter to be used for the Guardian nowait operation initiated by FTPquote_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Rename File on Remote Computer

The routines FTPrename and FTPrename_nw rename a file on a remote machine.

```
#include "ftpexth"
short FTPrename(short hd, short proxy, char *in,
                char *out)
short FTPrename_nw(short hd, short proxy, char *in,
                   char *out, long tag)
```

hd
file number.

proxy
1, this command applies to the proxy connection.

0, this command applies to the primary connection.

in
name of the file to be renamed

out
new file name.

tag
parameter to be used for the Guardian nowait operation initiated by FTPrename_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Retrieve the Reply Text

The routines FTPreplytext and FTPreplytext_nw retrieve the reply text.

```
#include "ftpexth"
short FTPreplytext(short hd, char  *buffer, short limit)
short FTPreplytext_nw(short hd, char * buffer,short limit,
                      long tag)
```

hd
: file number.

buffer
: buffer area to hold the result.

limit
: maximum number of bytes the buffer can hold. If reply text is longer than the limit, the error number is set to TRUNCATED,and data up to the limit size is returned in the buffer.

tag
: parameter to be used for the Guardian nowait operation Return value -1 if fails. The error number

The return value is -1 if the routine fails. The error number is set accordingly.

Clear Reply Queue

The routines FTPreset and FTPreset_nw clear the reply queue in order to synchronize with the remote FTP server.

```
#include "ftpexth"
short FTPreset(short hd, short proxy)
short FTPreset_nw(short hd, short proxy, long tag)
```

hd
: file number.

proxy
: 1, this command applies to the proxy connection.

  0, this command applies to the primary connection.

tag
: parameter to be used for the Guardian nowait operation initiated by FTPreset_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Remove Directory on Remote Machine

The routines FTPrmdir and FTPrmdir_nw removes the directory on a remote machine.

```
#include "ftpexth"
short FTPrmdir(short hd, short proxy, char *dir)
short FTPrmdir_nw(short hd, short proxy, char *dir,
                  long tag)
```

hd
: File number.

proxy
: 1, this command applies to the proxy connection.

  0, this command applies to the primary connection.

dir
: name of the directory to be removed.

tag
: parameter to be used for the Guardian nowait operation initiated by FTPrmdir_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Set File Code

The routines FTPsetfilecode and FTPsetfilecode_nw set the file code.

```
#include "ftpexth"
short FTPsetfilecode(short hd, short proxy,
                     short filecode)
short FTPsetfilecode_nw(short hd, short proxy,
                        short filecode, long tag)
```

hd
: File number.

proxy
: 1, this command applies to the proxy connection.

: 0, this command applies to the primary connection.

filecode
: file code to be used.

tag
: parameter to be used for the Guardian nowait operation initiated by FTPsetfilecode_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Set the Connection Timeout Value

The routines FTPsetcontimer and FTPsetcontimer_nw set the timeout value for accepting the connection from the server.

```
#include "ftpexth"
short FTPsetcontimer(short hd, short proxy,
                       short contimer)

short FTPsetcontimer_nw(short hd, short proxy,
                        short contimer, long tag)
```

hd
: File number.

proxy
: 1, this command applies to the proxy connection.

: 0, this command applies to the primary connection.

contimer
: a timeout value in seconds. Contimer is a number between 0 and 32767.

tag
: parameter to be used for the Guardian nowait operation initiated by FTPsetcontimer_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Set File Structure

The routines FTPsetstruct and FTPsetstruct_nw sets the structure of the file to File or Record structure.

```
#include "ftpexth"
short FTPsetstruct(short hd, short proxy,
                   char structure)
short FTPsetstruct_nw(short hd, short proxy,
                      char structure, long tag)
```

hd
: File number.

proxy

1, this command applies to the proxy connection.

0, this command applies to the primary connection.

structure

valid structure types include:

'F' File

'R' Record

tag

parameter to be used for the Guardian nowait operation initiated by FTPsetstruct_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Set Transfer Type

The routines FTPsettype and FTPsettype_nw set the transfer type to ascii or binary mode.

```
#include "ftpexth"
short FTPsettype(short hd, short proxy, char type)
short FTPsettype_nw(short hd, short proxy,
                    char type, long tag)
```

hd

File number.

proxy

1, this command applies to the proxy connection.

0, this command applies to the primary connection.

type

valid types include:

'A' Ascii

'I' Image

'T' Tenex

tag

parameter to be used for the Guardian nowait operation initiated by FTPsettype_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Display Remote Computer System Name

The routines FTPsystem and FTPsystem_nw return the system name of the remote machine.

```
#include "ftpexth"
short FTPsystem(short hd, short proxy,
                char *buffer, short limit)
short FTPsystem_nw(short hd, short proxy,
                   char *buffer, short limit,
                   long tag)
```

hd

File number.

proxy

1, this command applies to the proxy connection.

0, this command applies to the primary connection.

buffer

data buffer to hold the return data.

limit
maximum number of bytes the buffer area can hold.

tag
parameter to be used for the Guardian nowait operation initiated by FTPsystem_nw.

The return value is -1 if the routine fails. The error number is set accordingly.

Runtime Options

The routine FTPoption sets the runtime options for the FTP client.

```
#include "ftpexth"
void  FTPoption(char *ftp_option)
```

ftp_option
While opening a connection, if runtime options need to be specified, FTPoption() must be called before calling FTPopen(), FTPconnect(), FTPproxy(), and all no waited APIs.

Sample Program

This is a sample program that uses FTPAPI. The file is named FTPACIX.C.

```
/* FILE: FTPAPIX.C - FTPAPI Example - the FTPServer Address name, userid, password are hardcoded.

CISC Compile -

  C /IN FTPAPIXC, OUT $S.#FTPAPIX/ FTPAPIXO; WIDE, SYMBOLS,    RUNNABLE INSPECT, SEARCH "$SYSTEM.ZTCPIP.APILIB",
 SSV0    "$PERM.MMFTPAPI", SSV1 "$SYSTEM.ZTCPIP", SSV2 "$SYSTEM.SYSTEM"

RISC Compile -

  NMC /IN FTPAPIXC, OUT $S.#FTPAPIX/FTPAPIN; SYMBOLS, RUNNABLE, SEARCH "$SYSTEM.ZTCPIP.NAPILIB", SSV0
"$PERM.MMFTPAPI", SSV1 "$SYSTEM.ZTCPIP", SSV2 "$SYSTEM.SYSTEM"

*/

#include "ftpexth"
#include <stdio.h>
#include <errno.h>

extern int errno;

int main()

{
short hd1, nowait =0;
short hd2;

char buffer[1024];
char *cp;
int fd;
long tag=0;

char *opt="-d -s <subnet Address>";
/* Added for introducing FTPoption */
FTPoption(opt);

/* log messages directed to the terminal */
if ((hd1 = FTPopen("host1",-1,"user","passwd","","")) < 0)
{
printf("errno returned: %d\n",errno);
goto err;
}
printf("does this get printed immediately\n");
/* wait until I/O is completed */
for(; FTPio(&hd1,nowait,&cp,&tag) ==0;);
printf("open another connection\n");
/* log file directed to apilog */
if ((hd2 = FTPopen("host2",1234,"user","passwd","", "apilog")) < 0)
{
printf("errno returned: %d\n",errno);
goto err;
}
if (FTPproxy(hd1,"host3",-1,"user2","passwd","") < 0)
{
printf("errno returned: %d\n",errno);
goto err;
}
if (FTPpwd(hd1,PRIMARY,&buffer[0],511) < 0)
printf("errno returned: %d\n",errno);
else
printf("Pwd: %s\n",buffer);
```

```
if (FTPpwd(hd1,PROXY,&buffer[0],511)<0)
printf("errno returned: %d\n",errno);
if (FTPio(&hd1,WAIT,&cp,&tag))
printf("result: %s\n",cp);
else
printf("errno returned: %d\n",errno);
if (FTPget(hd1,PRIMARY,"abc","") < 0)
printf("errno returned: %d\n",errno);
else
printf("successful retrieve file abc\n");
if (FTPget(hd1,PROXY,"c","") < 0)
printf("errno returned: %d\n",errno);
else
printf("successful retrieve proxy file c\n");
if (FTPput(hd1,PRIMARY,"eft","mmm") < 0)

printf("errno returned: %d\n",errno);
else
printf("successfully store file eft onto remote host\n");
if (FTPrename(hd1,PRIMARY,"abc","mmm") < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully rename file from mmm to mmm1\n");
if (FTPlist(hd1,PRIMARY,"",buffer,511) < 0)
printf("LIST:errno returned: %d\n",errno);
else
printf("successfully list files: %s\n",buffer);
if (FTPlist(hd1,PROXY,"",buffer,1023) < 0)
printf("LIST:errno returned: %d\n",errno);
else
printf("successfully list proxy files: \n%s\n",buffer);
if (FTPappend(hd1,PRIMARY,"c","d") < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully append file c to d\n");
if (FTPmkdir(hd2,PRIMARY,"tdir") < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully mkdir tdir\n");
if (FTPmkdir(hd2,PRIMARY,"cdir") < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully mkdir cdir\n");
if (FTPrmdir(hd2,PRIMARY,"tdir") < 0)
printf("errno returned: %d\n",errno);
else
printf("successful rmdir tdir\n");
if (FTPsetfilecode(hd1, PRIMARY,100) < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully set file code\n");
if (FTPsetstruct(hd1, PRIMARY,'F') < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully set struct to F\n");
if (FTPsettype(hd1, PRIMARY, 'I') < 0)
printf("errno returned: %d\n",errno);
else
printf("successfully set type to I\n");
err:
(void)FTPbye(hd1,PRIMARY);
(void)FTPbye(hd2,PRIMARY);

}
```

For CISC objects, the command below compiles and links the sample program. The command asumes That the library files are in the standard location and that the the source file is on the compilation subvolume named $DATA.EXMPL.FTPAPIXC .

```
>C /IN FTPAPIXC, OUT $S.#FTPAPIX/ FTPAPIXO; WIDE, SYMBOLS, &
    RUNNABLE INSPECT, SEARCH "$SYSTEM.ZTCPIP.APILIB", SSV0 &
    "$PERM.MMFTPAPI", SSV1 "$SYSTEM.ZTCPIP", SSV2 &
    "$SYSTEM.SYSTEM"
```

For RISC objects, the command belo9w compiles and links the sample program.

```
>NMC /IN FTPAPIXC, OUT $S.#FTPAPIX/FTPAPIN; SYMBOLS, &
     RUNNABLE,  SEARCH "$SYSTEM.ZTCPIP.NAPILIB", &
      SSV0 "$PERM.MMFTPAPI", SSV1 "$SYSTEM.ZTCPIP", &
      SSV2 "$SYSTEM.SYSTEM"
```

For detailed information about compiling and linking programs in C or C++, refer to the *C/C++ Programmers Guide*.

△ **CAUTION:** When you compiling the C or C++ application and the obects are CISC objects, always specify the wide model.

Requirements

Each active FTP session must have a different log file to store the log information on disk. Currently, a log file that is also a disk file can not be shared among active FTP sessions. (If an active FTP session is already using “FTPlog” as its log file, any attempts by another FTP session to use “FTPlog” as its log file will be denied, and its output will be redirected to its terminal.)

The API Library relies on using the correct version of FTP. Whenever the API tries to activate an FTP session, one identity message is passed between the FTP client and the API, and if the identity message does not match, the session is aborted, and the error number is set accordingly.

An inactivity timer is included in the FTP client program. If the FTP client is initiated by the API, and there are no messages coming from the API to the FTP client for more than 900 seconds, the FTP program aborts. (The time-out interval is not programmable.)

There is no way to cancel the operation. If you do a CANCEL or AWAITIOX with timer specified and the timer expired. What it cancels is the interprocess communication to the FTP client, not its operation. Once it is canceled there is no way you can tell whether the operation is successful or not.

9 TFTP—Transferring Public Files

The TFTP client allows you to transfer public files to and from a remote host system. TFTP does not provide any mechanism for you to log on to the remote system and verify, through a user ID and password, which files you can access. The files you are allowed to retrieve from a remote system are typically secured for public access; that is, anyone on the network can read the files. The TFTP server on the remote system sets the restrictions on which files you can retrieve, as well as restrictions on storing files. (For an example of the types of restrictions placed on file access, see Section 10, Communicating With the TFTP Server.)

Running TFTP at a Terminal

To run the TFTP client at a terminal connected to a NonStop system, you enter the TFTP run command. The format of the command is:

```
tftp [ / run-option [ , run-option ] ... / ] [ host ] [port]
```

run-option

is an operating system RUN command option. See the RUN command in the *TACL Reference Manual* for a complete description of the run options.

host

sets a default remote host system to which transfer commands apply during the TFTP session, unless you specify some other remote system in a connect, get, or put command. You can specify *host* as a host name or host address. For information about specifying host names and addresses, see Addressing Remote Hosts on page 1-5, in Section 1, Introduction to TCP/IP Applications and Utilities,

port

sets a non default remote port for transfer.

The following command starts TFTP and sets the remote host to `dist101`:

```
TACL 9> tftp dist101

(The TFTP banner appears.)

tftp>
```

After you start TFTP, you can enter TFTP commands at the prompt. Table 8 provides a summary of the TFTP commands.

NOTE: Ask your system manager for the name of an appropriate TCP/IP process to serve as your transport service provider.

Considerations for choosing an appropriate process are:

Table 8 TFTP Command Summary

Command	Purpose
ascii	Set the file transfer mode to ASCII.
binary	Set the file transfer mode to binary.
connect	Specify the default remote host to which transfer commands apply.
quit	Exit TFTP.
get	Retrieve a file.
help *or* ?	Request local help information.

Table 8 TFTP Command Summary *(continued)*

Command	Purpose
mode	Set the file transfer mode.
put	Send a file.
rexmt	Set the time interval before retransmission of a packet that is not acknowledged by the remote system.
status	Display the current settings of the TFTP session and the file transfer parameters.
timeout	Set the timeout interval after which TFTP stops transmitting a packet that is not acknowledged by the remote system.
trace	Turns the display of packet tracing information on or off.
verbose	Turns the display of file transfer statistics on or off.

To enter one of these commands, you only need to specify enough characters to differentiate that command from all the other commands. For example, you can enter the letter c for the connect command:

```
tftp> c medlabs
```

Setting Session and File Transfer Parameters

You can specify settings for parameters that control the way files are transferred and the displays that appear during your session. In the following example, the first two commands set the file transfer mode to binary and turn on the display of file transfer statistics. The next two commands set up the way in which TFTP retransmits packets that are not acknowledged by the remote system. The rexmt command specifies that TFTP will retransmit a packet if it is not acknowledged by the remote system within 10 seconds. The timeout command specifies that TFTP stop retransmitting the packet 40 seconds after the first transmission, even though the packet is unacknowledged:

```
tftp> mode binary
tftp> verbose
Verbose mode on.
tftp> rexmt 10
tftp> timeout 40
```

Each parameter has a default value or setting (on or off) when you start TFTP. for detailed information, see “TFTP Command Reference” (page 121).

Sending and Retrieving Files

To send files, you use the put command; to retrieve files, you use the get command. You can transfer one or more files using a single command. A file transfer operation copies the original file to or from the remote system.

You can specify the remote system in one of three ways:

- Specify the remote system when you enter the TFTP run command. This method is convenient if you intend to transfer many files to or from the same remote system during your TFTP session.
- Enter a connect command. You should use the connect command if the remote system is not accessible through the well-known port (UDP port 69). This command allows you to specify a port.
- Enter the remote host name preceding the remote file name in a get or put command. This method overrides a default remote system if you have specified one.

You might not be able to transfer a file due to the security restrictions placed on files at the remote system. In this situation, ask the administrator or manager of that system what security restrictions apply.

TFTP Command Reference

The following pages contain descriptions of the syntax (enclosed in boxes) and rules for using TFTP commands, and provide examples of typical ways to use the commands.

ascii

Use the ascii command to set the file transfer mode used for data transfer and storage to ASCII.

```
ascii
```

You can also use the mode command to perform this function.

Example

Assume the verbose toggle is on. To change the file transfer mode to ASCII, enter the following command:

```
tftp> ascii
mode set to netascii
```

binary

Use the binary command to set the file transfer mode used for data transfer and storage to binary, which is also called octet.

```
binary
```

You can also use the mode command to perform this function.

Example

Assume the verbose toggle is on. To change the file transfer mode to binary, enter the following command:

```
tftp> binary
mode set to octet
```

connect

Use the connect command to specify the remote system to be used for file transfers when you do not specify a remote host in a get or put command.

```
connect host [ port]
```

host

is a host name or host address identifying the remote system. See Addressing Remote Hosts on page 1-5 in Section 1, Introduction to TCP/IP Applications and Utilities.

port

specifies the number of a UDP port on the remote system that TFTP will use for the file transfers. If you omit *port*, the default UDP port is 69.

TFTP does not establish a connection when you enter the connect command. The connect command only sets the default remote system to be used.

Example

To set the default remote system to dynamo and the port to 68, enter the following command:

```
tftp> connect dynamo 68
```

get

Use the get command to copy one or more remote files to the local system.

```
get { [host:]remote-file [ [host:]local-file ] }

    {                                  }

    { file1 file2 ... filen            }
```

host
is a host name identifying the remote system from which the file is copied. You cannot specify a host address. For information about specifying host names and addresses, see Addressing Remote Hosts on page 1-5, in Section 1, Introduction to TCP/IP Applications and Utilities.

remote-file
specifies the name of the remote file to be copied. Specify *remote-file* as required by the remote system.

If you specify only the name of the remote file and not its full pathname, the TFTP server assumes that the full pathname for the file is \system.$volume.subvolume.remote-file, where \system.$volume.subvolume is the first configured subvolume for TFTPSRV.

local-file
specifies a name for the local file. Follow the Guardian file naming conventions. Unless you qualify the file name, the file is stored in your current default subvolume.

If you omit *local-file*, TFTP uses *remote-file* for the new file name. If *remote-file* is not a valid Guardian file name, you must specify *local-file*.

filen
specifies one of a set of remote files to be copied. You cannot include a host name when specifying a list of remote files. Each local file is assigned the same name as its corresponding remote file.

The list should include only remote file names that can be valid Guardian local file names. You must transfer other files one at a time and specify a local file name for each one.

You cannot copy a file unless the remote system allows access through its TFTP server.

Examples

In the following example, a file named typetips is copied from a remote system named ssu and assigned the local file name typetips:

```
tftp> get ssu:typetips
```

To copy the same file and assign the name goodtips, enter the following command:

```
tftp> get ssu:typetips goodtips
```

In the next example, three files are copied from the remote system named asulab. The local files are assigned the same names as the remote files.

```
tftp> connect asulab
tftp> get stocks gameplan refs
```

help

Use the help command to display either a list of all commands or helpful information about a specific command.

```
help [ command-name ]
```

command-name

specifies a TFTP command. If you omit *command-name*, a list of all TFTP commands is displayed.

Example

In the following example, the help command requests information about the connect command:

```
tftp> help connect
connect to remote tftp
```

mode

Use the mode command to set the file transfer mode used for data transfer.

```
mode { [net]ascii | image | binary | octet }
```

[net]ascii

specifies that data is to be transferred and stored in ASCII format. This is the default mode.

image | binary | octet

specifies that data is to be transferred and stored in binary format.

You can also use the ascii and binary commands to set the file transfer mode.

Examples

Assume the verbose toggle is on. The following examples show three ways to enter the mode command:

```
tftp> mode netascii
mode set to netascii
tftp> mode binary
mode set to octet
tftp> mode ascii
mode set to netascii
```

put

Use the put command to copy one or more local files to the remote system.

```
put { local-file [ [host:]remote-file ]                }
    {                                                  }
    { file1 file2 ... filen [host:]remote-directory }
```

local-file

specifies the name of the local file to be copied. Follow the Guardian naming conventions. If you qualify the file name when transferring the file to an operating system other than the Guardian system, you must specify a remote file name.

host

is a host name identifying the remote system. You cannot specify a host address. See Addressing Remote Hosts on page 1-5, in Section 1, Introduction to TCP/IP Applications and Utilities.

remote-file

specifies a name for the remote file. Specify *remote-file* as required by the remote system. If you omit *remote-file*, TFTP uses the *local-file* name for the new file.

If you specify only the name of the remote file and not a full pathname, the TFTP server assumes that the full pathname for the file is \system.$volume.subvolume.remote-file, where \system.$volume.subvolume is the first configured subvolume for TFTPSRV.

If a remote file having the file name you specify already exists, the PUT operation succeeds only if that remote file is secured xNxx.

filen

specifies one of a set of local files to be copied. Each remote file is assigned the same name as its corresponding local file. The files are copied from the current default subvolume unless you specify qualified Guardian names.

If you qualify a local file name, the system, volume, and subvolume parts of the name are omitted from the corresponding remote file name, and the remote file is stored in the current directory. For example, assume the remote directory is currently /etc/test. The following command creates the remote files /etc/test/x1 and /etc/test/x2:

```
tftp> put $vol1.subva.x1 $vol2.subvb.x2
```

[*host*:]*remote-directory*

specifies a directory and, optionally, the host name on the remote system. The local files are copied to the specified directory.

To use this form of the put command, you must transfer files to a UNIX system. TFTP assumes you are specifying a UNIX directory name.

You cannot copy a file unless the remote system allows access through its TFTP server.h3

Examples

In the following example, a file named newcures is copied to a remote system named medlab and assigned the remote file name newcures:

```
tftp> connect medlab

tftp> put newcures
```

To copy files named stats87 and stats88 to the statistics directory on the medlab system, you enter the following command:

```
tftp> put stats87 stats88 medlab:statistics
```

In the next example, a local file named topbooks is copied to the remote system named asulab and assigned the name booklist:

```
tftp> put topbooks asulab:booklist
```

quit

Use the quit command to exit TFTP.

```
quit
```

You can also press CTRL/Y to exit TFTP.

Example

```
tftp> quit
```

rexmt

Use the rexmt command to set the number of seconds TFTP should wait for a packet to be acknowledged by the remote system before retransmitting the packet.

```
rexmt interval
```

interval

specifies the number of seconds to wait before retransmitting a packet. The default value is 5 seconds.

Example

The following command sets the retransmission interval to 3 seconds:

```
tftp> rexmt 3
```

If you omit *interval*, TFTP prompts you for a value, as in the following example:

```
tftp> rexmt
(value) 3
```

status

Use the status command to display the current settings of the TFTP session and the file transfer parameters.

```
status
```

Example

The status command displays the following information:

```
tftp> status
Connected to medlab.MRAM.COM
Mode: netascii Verbose: off Tracing: on
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp>
```

timeout

Use the timeout command to set the total number of seconds TFTP continues to retransmit a packet that is not acknowledged by the remote system.

```
timeout total-interval
```

total-interval

specifies the number of seconds from the first transmission of a packet to the last retransmission attempt. The default interval is 25 seconds.

If a packet is not acknowledged by the remote system, TFTP retransmits it in intervals specified by the rexmt command until the seconds specified by the timeout command have been exceeded.

Example

The following command sets a timeout interval of 50 seconds:

```
tftp> timeout 50
```

If you omit *total-interval*, TFTP prompts you for a value, as in the following example:

```
tftp> timeout
(value) 30
```

trace

Use the trace command to turn the display of packet tracing information on or off. This command operates as a toggle.

```
trace
```

When you start TFTP, the trace toggle is off. If you turn the trace toggle on, TFTP displays information about each packet as it is sent or received.

Example

Assume the verbose toggle is on. To change the setting of the trace toggle, enter the following command:

```
tftp> trace
Packet tracing on.
```

In trace mode, the following type of information appears:

```
tftp> get /stats/weeksummary weeksum
sent RRQ <file=/stats/weeksummary, mode ascii>
received DATA <block=1 512 bytes>
```

```
sent ACK <block 1>
     .
     .
```

verbose

Use the verbose command to turn the display of file transfer statistics on or off. This command operates as a toggle.

```
verbose
```

When you start TFTP, the verbose toggle is off.

If you turn verbose mode on, TFTP displays the number of bytes sent or received, and the number of seconds required for the operation.

NOTE: The statistics displayed by this command are approximate and should not be used to compare FTP performance with other implementations.

Example

The following example shows how to turn verbose mode on and indicates the type of information that appears:

```
tftp> verbose
Verbose mode on.
tftp> get weeksum
Received 34754 bytes in 2 seconds
tftp> put telelist
Sent 9765 bytes in 1 second
tftp>
```

?

Use the ? command to display either a list of all commands or helpful information about a specific command. (The ? command performs the same operation as the help command.)

```
? [ command-name ]
```

command-name

specifies a TFTP command. If you omit *command-name*, a list of all TFTP commands is displayed.

Example

In the following example, the ? command requests information about the rexmt command:

```
tftp> ? rexmt
set per-packet retransmission timeout
```

10 Communicating With the TFTP Server

HP NonStop implementations of TCP/IP provide a TFTP server that allows you to transfer files to and from a NonStop system while using a TFTP client on another network system.

The TFTP Server can handle as many as 120 simultaneous requests without performance degradation

NOTE: The TFTP Server consists of two distinct process types: TFTPSRV and TFTPCHLD. A single TFTPSRV process validates requests and one or more TFTPCHLD processes handle data transfer. For every sixteen TFTP requests, TFTPSRV generates a new TFTPCHLD process. An individual TFTPCHLD process terminates when it has handled sixteen requests and no further request is pending or when it has been idle for 10 minutes.

The TFTP server imposes restrictions on the files that can be transferred.

TFTP Server Codes

The TFTP server supports TFTP packets with the operating codes described in Section . The TFTP server does not include TFTP mail services.

Table 9 TFTP Codes Supported

Code	Meaning
ACK	Acknowledgment that a data packet has been received.
DATA	The packet contains a block of data.
ERR	The packet contains an error code and message.
RRQ	Read request indicating the file to be read and the mode.
WRQ	Write request indicating the file to be written and the mode.

Securing Files in the $SYSTEM.CSSnn Subvolume (SWAN Users)

For the TFTP server to function, SWAN users must make sure that all files in the $SYSTEM CSSnn subvolume are secured for network read access (Nxxx). recommends that SWAN users secure these files for both network read access and network execute access (NxNx). For more information on file security codes, see Table 10 (page 128) in "Security Code Restrictions" (page 128).

Binding to a Single Subnet

Before starting the TFTP Server process, the operator can set the TACL param TFTP^HOST^IP to bind the TFTP Server to a single, specified subnet.

For example, to bind the TFTP server to IPv4 address 192.168.10.10, the operator would issue the following command:

```
PARAM TFTP^HOST^IP 192.168.10.10
```

TFTP^HOST^IP also accepts IPv6 addresses. For example, to bind the TFTP server to IPv6 address `3ffe:1200:214:1:a00:8eff:fe04:6ef2`, the operator would issue the following command:

```
PARAM TFTP^HOST^IP 3ffe:1200:214:1:a00:8eff:fe04:6ef2
```

When a subsequent TACL RUN command is issued, the TFTP Server binds to port 69 of the specified subnet.

If the operator does not use PARAM TFTP^HOST^IP, the TFTP Server binds to all the subnets. As a result:

- No free ports are available for the operator to configure other applications
- The TFTP Server's performance will degrade.

For an example of the TFTP Server RUN command, see "Subvolume Restrictions" below.

NOTE: Ask your system manager for the name of the TCP/IP process on your system. If the process is named anything other than $ZTC0, specify that name in a TCPIP^PROCESS^NAME TACL PARAM command before running tftpsrv.

Refer to the *TACL Reference Manual*for more information about TACL PARAM commands.

You might also need to resolve the DEFINE name =TCPIP^HOST^FILE.

Transfer Restrictions

Because TFTP services do not require you to provide a user ID or password, the TFTP server places restrictions on the types of files you can retrieve from and store on a NonStop system.

Subvolume Restrictions

The user or operator who starts the TFTP server can specify from one to ten subvolumes in the RUN command. If subvolumes are specified, all files transferred by a TFTP client to the NonStop system must be stored in one of the specified subvolumes, and only files in those subvolumes can be retrieved by TFTP clients. For example, the following command specifies that all TFTP transfers will operate on files in the subvolumes $NET.PUB and $GRP.MEMOS:

```
TACL 5> run $system.ztcpip.tftpsrv /nowait, pri 150,&
out $null/ $net.pub $grp.memos
```

In this example, the run options that appear between the slashes (/) are the recommended options for starting the TFTP server. The options specify that the current TACL process will not suspend itself while the new process runs, that the execution priority is 150, and that the output goes to $NULL.

NOTE: The user or operator who starts the TFTP server must have super ID capabilities.

If no subvolumes are specified, TFTP can only transfer files to or from the subvolume named $DATA.PUBLIC. For more information about starting the TFTP server, see the *TCP/IP Configuration and Management Manual*.

Security Code Restrictions

A file is protected in the Guardian file system by a security code that specifies who can read, write, execute, and purge the file. The TFTP server examines the security code before allowing a file to be transferred to the TFTP client or replaced by a file sent from the client. The form of the security code is RWEP, indicating the following privileges:

Table 10 File Security Codes

R	Read access
W	Write access
E	Execute access
P	Purge access

In the security code, these privileges are granted to certain types of users; for example, the security code NGGG grants read access to N (all network users) and write, execute, and purge access to G (group members).

For a complete description of file security codes, see the *Guardian User's Guide*.

Restrictions on Retrieving Files

You can retrieve only those files that have a security code of Nxxx (where x can be any valid security code. The file must be secured with the read access privilege granted to any network user (N). The TFTP server does not consider the other three types of access when you retrieve a file.

Restrictions on Storing Files

When using the PUT command, you can store files on a remote system if you observe the following file security restrictions:

- To overwrite an existing remote file with a local file, you must make sure that the file on the remote system is secured for write access granted to the network user (xNxx).

  NOTE: Both the local file and the remote file you want to overwrite must be in the same mode (for example ASCII). Otherwise, the operation will fail, and you will receive an error message.

- If you transfer a local file to a remote system and store it in a newly created file on that system, the TFTP server gives the newly created file a security code of NUNU.

  NOTE: You can create a new file in any of the subvolumes specified when the TFTP server was started. If no subvolumes were specified, you can create new files only in $DATA.PUBLIC and only if $DATA.PUBLIC is present on the remote system.

11 TELNET—Using a Network Virtual Terminal

The TELNET application allows you to emulate a virtual terminal connected to a remote host. You can connect to any remote host on the network that has a TELNET server. You can also transfer data to a remote system through a non-TELNET port.

Running TELNET at a Terminal

To run the TELNET client on your Guardian system, you must enter the TELNET run command at the command interpreter prompt. The form of the TELNET run command is:

```
telnet [ / run-option [ , run-option ] ... / ]

       [ host [ port ] ]
```

run-option
: is an operating system RUN command option. See the RUN command in the *TACL Reference Manual* for a complete description of the run options. If you specify an IN file, the file must be a terminal.

host
: identifies the remote host system. You can specify *host* as a host name or host internet address. For information about specifying host names and addresses, see Addressing Remote Hosts on page 1-5, in Section 1, Introduction to TCP/IP Applications and Utilities.

port
: identifies the TCP port number. The default (well-known) port number is 23.

If you do not specify *host*, TELNET places you in command mode and displays the TELNET prompt (telnet>). You can enter TELNET client commands at this prompt.

If you specify *host*, TELNET connects you to the remote system and places you in input mode. TELNET negotiates with the remote system to determine what terminal options are available. After TELNET completes the negotiations, prompts from the remote system should appear.

NOTE: Ask your system manager for the name of an appropriate TCP/IP process to serve as your transport service provider.

Considerations for choosing an appropriate process are:

While communicating with the remote system, you can enter the escape character to invoke the telnet> prompt and to change to TELNET command mode. You enter the escape character by pressing the CTRL key simultaneously with the right bracket (]) key.

After you enter a TELNET client command and press the RETURN key, you return to input mode.

Table 11 summarizes the TELNET client commands.

Table 11 TELNET Command Summary

Command	Purpose
close	Disconnect from remote system and return to TELNET prompt.
display	Display TELNET special characters and current settings of toggle controls.
help *or* ?	Display information about TELNET commands.
mode	Set the input mode.
open	Connect to remote system.

Table 11 TELNET Command Summary *(continued)*

Command	Purpose
quit	Disconnect from remote system and exit TELNET.
send	Send special characters to remote system.
status	Display current status of TELNET.
toggle	Turn TELNET toggle controls on or off.
ttywritesz	Specify number of characters to be sent at one time from remote system to your terminal.

To enter one of these commands, you only need to specify enough characters to differentiate the command from all the other commands. For example, you can enter o for the open command:

```
telnet> o mainsys
```

You can also abbreviate arguments of the display, mode, set, and toggle commands; however, you must make each argument distinct from all other arguments for the command you are entering. For example, you can abbreviate localchars in the toggle command by entering the letter l as follows:

```
telnet> toggle l
```

If you are communicating with the NonStop TELNET server, a banner appears, indicating the services available. If the system indicates that the service you select is not available, ask the system manager of the remote system to start the service.

Connecting to a Remote System

To establish a connection to a remote system, you can either specify the host name or address in the TELNET run command, or you can enter an open command at the telnet> prompt. For example, the following command connects to a system named salesctr:

```
telnet> open salesctr
```

To disconnect from the remote system, you log off.

NOTE: When the NonStop TELNET client connects with a TELNET server, it asks the server if it recognizes terminal subtype. Many UNIX systems support a terminal capability database through a TERMCAP file.

Displaying Information About Your TELNET Session

The display and status commands provide information about your TELNET operating environment. You can use the status command to display the name of the remote system, the input mode, and the escape character. Use the display command when you want to know the current setting of a toggle or value of a special character, such as the erase character or the interrupt character.

Controlling the TELNET Environment

You can use the toggle command to control the way TELNET responds to events. The localchars toggle controls the recognition of the following special characters: erase, flushoutput, interrupt, kill, and quit. The initial setting of the localchars toggle is off. When localchars is on, you can enter a special character to send the related TELNET sequence to the remote system. Table 11-2 lists the special characters you can use with TELNET, indicating the ones affected by the localchars toggle. A circumflex (^) indicates that you press the CTRL key simultaneously with the other character shown.

Table 12 Special Characters

Name	Keys	Function
echo	^E	In line input mode, turns on or off the display (echoing) of characters you are entering. This character is useful when you are entering a password.
eof	^Y	In line input mode, sends this character to the remote system (if eof is the first character entered on the line).
erase	^H	In character input mode with the localchars toggle on, sends the TELNET EC (erase character) sequence to the remote system.
escape	^]	Sends the TELNET escape character. You use these keys to invoke the TELNET prompt.
flushoutput	^O	With the localchars toggle on, sends the TELNET AO (abort output) sequence to the remote system.
interrupt	^C	With the localchars toggle on, sends the TELNET IP (interrupt processing) sequence to the remote system.
kill	^X	In character input mode with the localchars toggle on, sends the TELNET EL (erase line) sequence to the remote system.
quit	^ \	With the localchars toggle on, sends the TELNET BRK (break) sequence to the remote system.

In the following example, the localchars toggle is turned on, and the flushoutput character is sent to the remote system:

```
remsys:

(Enter escape character.)

(Set localchars on.)

telnet> toggle localchars
remsys: dir
      .
      .

(Enter CTRL/O to flush the output.)

remsys:
```

Using TELNET on Other Ports

If you request a connection to a port that does not have a TELNET server, TELNET does not negotiate with the remote system. TELNET does, however, provides a mechanism for connecting to a remote server for data transfer. For example, assume that you try to establish a TELNET connection to a host system named offc412 on port number 21, which is usually the well-known port for the FTP server. The result is:

```
TACL 4> telnet offc412 21
Trying...  Connected to offc412
Escape is ''
```

```
---> 220 offc412.zzzco.COM   FTP server ...
USER matthew
---> 331 password required for matthew
PASS bday
---> 230 user matthew logged in
     .
     .
```

The remote FTP server sends the lines that begin with --->. In this case, the TELNET client communicates with the FTP server.

TELNET Command Reference

The following pages contain descriptions of the syntax (enclosed in boxes) and rules for using TELNET commands, and provide examples of typical ways to use the commands.

close

Use the close command to disconnect from the remote system and return to the TELNET prompt at the local system.

```
close
```

Example

Assume you want to end a session with a system that prompts you for another logon ID after you enter a command to log off. To disconnect from the system, you use the following technique:

```
(Enter command to log off.)

login:

(Enter escape character.)

telnet> close
telnet>
```

display

Use the display command to display the values for special characters and the current setting of toggle controls.

```
display [ argument ] ...
```

argument

specifies the element that you want to display; *argument* can be any one of the following:

autoflush	autosynch	crmod
echo	eof	erase
escape	flushoutput	interrupt
kill	localchars	netdata
options	quit	

You only need to specify enough characters to uniquely identify the argument.

If you do not specify any arguments, the current setting for all arguments is displayed. In the display, a circumflex (^) represents the CTRL key.

Example

To display the interrupt character and the current setting of netdata, enter the following display command:

```
telnet> display i n
[^C]  interrupt.
Won't print hexadecimal representation of network traffic.

(Press RETURN to return to terminal mode.)

TACL 10>
```

help

Use the help command to display either a summary of all TELNET commands or information about a specific command.

```
help [ command-name ]
```

command-name

specifies the name of the command you want described.

Examples

To request information about the open command, enter the following command:

```
telnet> ? open
connect to a site
```

See the ? command for more examples.

mode

Use the mode command to specify the input mode you want to use.

```
mode { c[haracter] | l[ine] }
```

c[haracter] | l[ine]

specifies the type of input.

In character mode, TELNET immediately sends each character you type to the remote system. In line mode, the text you type is echoed locally and sent to the remote host when you press the RETURN key.

TELNET enters the mode you specify only if the remote host is capable of entering that mode.

By default, the mode is character.

Example

To change to line mode, enter the following command:

```
telnet> mode l
```

open

Use the open command to establish a connection to a remote system.

```
open host [ port ]
```

host

is a host name or host address identifying the remote system. For information about specifying host names and addresses, see Addressing Remote Hosts on page 1-5, in Section 1, Introduction to TCP/IP Applications and Utilities.

port

specifies the number of a port on the remote system that you want TELNET to contact. If you omit *port*, TELNET attempts to contact a TELNET server at the default (well-known) port, which is 23.

If you connect to a non-TELNET port, TELNET connects to a remote server on the port and allows you to transfer data. For more information on how to connect to non-default ports, see "Using TELNET on Other Ports" (page 132).

Example

The following command establishes a connection to the TELNET server on a system named dist101:

```
telnet> open dist101
```

quit

Use the quit command to close the remote connection and exit TELNET.

```
quit
```

If TELNET encounters the end of the file, the result is the same as issuing a `quit` command.

Example

To disconnect from the remote system and exit TELNET, enter the following command:

```
telnet> quit
```

send

Use the send command to send one or more special character sequences to the remote system.

```
send { argument [ argument ] ... }
     { ?                         }
```

argument

is the name of the character sequence you want to send. The arguments and their corresponding TELNET sequences are described in Table 13.

?

specifies that you want to view information about the send command.

The result of sending a character sequence depends on the capabilities of the remote system. If the sequence has no significance for the remote system, it may not respond to your request.

Table 13 Send Command Arguments

Argument	TELNET Sequence Sent
ao	AO (abort output) sequence, which asks the remote system to discard from the remote system all output currently prepared for your terminal (flush all output).
ayt	AYT (are you there) sequence.
brk	BRK (break) sequence.
ec	EC (erase character) sequence, which asks the remote system to erase the last character you entered.
el	EL (erase line) sequence, which asks the remote system to erase the line that you are currently entering.
escape	Current TELNET escape character. The default escape character is CTRL/] (^]).
ga	GA (go ahead) sequence, which signals the remote system that it can send data. This sequence is provided for half-duplex terminals, but is not required for most systems.
ip	IP (interrupt process) sequence, which asks the remote system to abort the currently running process.
nop	NOP (no operation) sequence.
synch	TELNET SYNCH sequence, which causes the remote system to discard all previously typed input that has not been read yet. If the synch sequence is not significant to the remote system, a lowercase letter (r) might appear on your terminal.

Example

Assume you are working with a remote system in character input mode, and you want to erase the characters you have entered on a line. In this example, the remote system prompt is a dollar sign ($):

```
$ cp outk Enter escape character.
telnet> send el Send erase line sequence.
$ Remote system erases line.
```

status

Use the status command to display the name of the remote system to which you are connected, the current input mode, and the current escape character.

```
status
```

Example

To display the status, enter the following command:

```
telnet> status
Connected to dist100.
Operating in character-at-a-time mode.
Escape character is '^]'.
```

toggle

Use the toggle command to turn on or off toggles that control how TELNET responds to events.

```
toggle { argument [ argument ] ... }
       { ?                         }
```

argument

specifies the name of the toggle control you want to change. You can abbreviate the name to the letters required to uniquely identify the toggle.Table 14 describes each toggle and its initial setting when you start TELNET.

?

requests information about the `toggle` command.

Table 14 Toggle Controls

Argument	Initial Setting	Function
autoflush	on	Determines whether output is flushed when you send interrupt (intr) characters. If both autoflush and localchars are on when you enter the ao, intr, or quit character, TELNET does not display any data until the remote system acknowledges (with a TELNET Timing Mark option) that it has processed the TELNET sequence you sent.
autosynch	off	Determines whether the TELNET SYNCH sequence is sent. If both autosynch and localchars are on when you enter an intr character, TELNET sends the SYNCH sequence and previously typed data is flushed.
crmod	off	Controls carriage return mode (crmod). When crmod is on, carriage return

Table 14 Toggle Controls *(continued)*

Argument	Initial Setting	Function
		characters received from the remote system are mapped into a carriage return followed by a line feed (unless a line feed is sent also). This mode is useful only if the remote system sends carriage returns but not line feeds.
localchars	off	Controls the local recognition of special characters and their transformation into appropriate TELNET sequences. When it is on, you can use the erase, flushoutput, interrupt, kill, and quit characters to send the related TELNET sequence to the remote system.
netdata	off	When netdata is on, all network data is displayed in hexadecimal format. This toggle is useful for debugging.
options	off	When options is on, internal TELNET protocol processing that relates to TELNET options is displayed. This toggle is useful for debugging.

Example

To change the setting of the autoflush and localchars toggles, enter the following command:

```
telnet> toggle autof l
```

ttywritesz

Use the ttywritesz command to determine the maximum number of characters that can be sent at one time to your terminal from the remote system.

`ttywritesz` *`argument`*

`argument`

is an integer specifying the size of your terminal buffer, in bytes. The default size is 70 bytes.

Example

To specify a buffer size of 75 bytes, enter the following command:

```
telnet> ttywritesz 75
```

?

Use the ? command to display either a summary of all TELNET commands or information about a specific command. (This command performs the same function as the help command.)

`?` `[` *`command-name`* `]`

`command-name`

specifies the name of the command you want described.

Examples

To request information about the `toggle` command, enter the following command:

```
telnet> ? toggle
toggle operating parameters ('toggle ?' for more)
```

To display information about each toggle, enter `toggle ?` as indicated in the display shown above:

```
telnet> toggle ?
autoflush            toggle flushing of output when send...
autosynch            toggle automatic sending of interrupt...
crmod                toggle mapping of received carriage...
           .
           .
options (debugging)  toggle viewing of options processing
?                    display help information
telnet>
```

12 Communicating With the TELNET Server

You can establish communications with the HP NonStop TELNET server from any system on the network that has a TELNET client. You invoke the TELNET client at a terminal on the other system, and the client establishes the connection with the TELNET server.

Establishing a Connection

When you request a connection to a NonStop system from your TELNET client, the TELNET server performs the following operations:

1. Allocates an emulated terminal subdevice for the client; for example, `$ZTNT.#PTY05`.
2. Sends to the client a welcome banner that indicates the system name on which the server is running and the terminal name.
3. Negotiates TELNET options with the client.
   a. Displays on your terminal a list of all available services. (These services are made available by the system manager of the server system.) After you select a service, the program that provides the service begins executing.
4. Starts a TACL process for you, if the system manager of the server system has not configured any services.

Assume that you have connected to a system named mktg. The following information appears on your screen, and you can start a TACL process to access the system:

```
WELCOME TO mktg.Tandem.COMM...
T9553c20  TELNET SERVER ..........

Available Services:

TACL  EXIT
Enter Choice> tacl
       .
       .
TACL 1>
```

See the Guardian User's Guide for more information about using TACL.

Requesting Block Mode

If the TELNET client uses sub-option TELOPT_TTYPE during negotiation and sends one of the following strings, the TELNET server sets the 653X emulation mode, which includes block mode:

```
TN6530    TN653X    6530    653X
```

Otherwise, the terminal is set to the normal TELNET network virtual mode only. In block mode, more applications are available to the terminal user; for example, TEDIT on a NonStop system operates in block mode.

The NonStop server ignores all other terminal-type options requested during negotiation with a client program.

Available TELNET Options

Table 15 describes the TELNET options supported by the TELNET server on NonStop systems. The TELNET server does not support any other options described in the Internet Request for Comments (RFC) 854. Note that the sub-option TELOPT-TTYPE is ignored except as described in “Requesting Block Mode” (page 139).

Table 15 TELNET Options Supported

Option	Resulting Action
AO (Abort Output)	Discards all pending output and synchronizes the systems.
AYT (Are You There)	Sends the message "[Yes]" to the client.
BREAK (Send Break)	Sends a BREAK message to the break owner of the associated terminal.
DM (Data Mark)	Causes the client and server to synchronize with each other.
DO (Do option)	Signifies that either the client or the server is asking the other to use a certain option.
DONT (Dont option)	Signifies that either the client or the server is requesting the other not to use a certain option that the other was willing to use.
EC (Erase Character)	Erases the last character entered, unless that character has already been consumed by a process that is using the associated terminal.
EL (Erase Line)	Erases the current line being entered. The result is the same as typing a line-kill character.
IP (Interrupt Process)	Sends a BREAK message to the break owner of the associated terminal.
SB (Subnegotiation)	Starts the negotiation of a sub-option. This option is recognized but ignored.
WILL (Will option)	Signifies that either the client or the server is willing to use a certain option.
WONT (Wont option)	Signifies that either the client or the server will not use an option that the other asked it to use.

The options negotiated using `DO`, `DONT`, `WILL`, and `WONT` are described in Table 16:

Table 16 TELNET Options Negotiable Using DO, DONT, WILL, and WONT

Option	Resulting Action
BINARY (Binary mode)	Signifies whether data is to be passed to the user process with or without being interpreted.
ECHO (Echo characters)	Signifies whether the client or server is willing or unwilling to echo characters.
SGA (Suppress Go Ahead)	Signifies whether the Go Ahead protocol is or is not suppressed. (This option is used mainly for half-duplex terminals.)

13 Using Your Workstation as a 6530 Terminal

The TN6530 is a multiple-page terminal emulation utility that runs on a BSD-based UNIX system (specifically, the Sun Microsystems machine running SunOS Release 3.5 or higher, or UNIX 4.2). When you run TN6530, your workstation emulates a 6530 series (653X) terminal, allowing you to log on to a NonStop computer and run applications that depend on the full capabilities of a 653X terminal.

TN6530 provides access to a NonStop system through the TELNET protocol on a TCP/IP network.

While using TN6530, you can easily change from workstation mode to emulator mode.

NOTE: Before the TN6530 can be used, the software must be installed. To use conversational mode or 653X block-mode functions, your window size must be at least 80 columns by 25 rows. Appendix C contains detailed information about installing TN6530.

Starting TN6530

To use TN6530, you must have the program files installed on your workstation. You must have a logon name (Guardian user ID) on the host system you want to use. The system manager of the host system can create a user ID for you.

The form of the TN6530 run command is:

```
tn6530 [ host [ port ] ]
```

host

identifies the remote NonStop host system. You can specify *host* as a host name or host internet address. See "Addressing Remote Hosts" in Section 1, Introduction to TCP/IP Applications and Utilities, for information about specifying host names and addresses.

port

identifies the TCP port number. The well-known (default) port is 23.

To start the program, follow these steps:

1. At the UNIX prompt, type tn6530, followed by the name or internet address of the host system you want to use. For example, to connect to a host named warehs1, enter the following command and press the ENTER key (or the RETURN key):

   ```
   % tn6530 warehs1

   Trying...Connected to warehs1
   Escape character is '^]'
   ```

   You can request a specific port by specifying its number following the host name (or address), as in the following example:

   ```
   % tn6530 warehs1 20.
   ```

   You should receive a welcome banner from the host system, indicating its system name and a terminal name. The host system also displays a list of available services. Here is an example welcome banner:

   ```
   WELCOME TO warehs1.Tandem.COM...
   T 9553 C20 TELNET SERVER ....

   Available Services:

   TACL  EXIT
   Enter Choice>
   ```

2. Choose the service you want to use by typing the service name and pressing the RETURN key. For example, to start a TACL process enter the following choice:

```
Enter Choice> tacl

TACL 1>
```

NOTE: If you receive a message indicating that the service you have selected is not available, inform the system manager of the remote system that the service has not been started.

If you omit the host name (or address) when you start TN6530, or if your connection to the host system does not succeed, you can use the TELNET open command to try to make the connection again. TN6530 allows you to use all the commands and options provided by the TELNET client; see Section 11, TELNET—Using a Network Virtual Terminal, for a description of these commands and options.

3. Log on to the system.

```
TACL 1> LOGON ADMIN.BONNIE

(Welcome message and other system messages appear.)

TACL 2>
```

You must have a logon name consisting of a group name and a user name, such as ADMIN.BONNIE in this example, and a password. You can change the password using the TACL PASSWORD command.

NOTE: To use 653X block-mode functions, you must start TN6530 in a window that is at least 80 columns by 25 rows.

Suspending TN6530

Use the following procedure to temporarily exit your host system and return to TN6530 command mode:

1. Press CTRL/] (CONTROL key and close bracket key, simultaneously). This is the escape sequence indicated by ^] in the first message you see after connecting to the host system:

```
TACL 5>

(Press CTRL/])

tn6530>
```

After you execute a TELNET command, you automatically return to the host system prompt; in this example, TACL 5>.

2. If you want to return to UNIX temporarily, type the letter z at the TN6530 prompt:

```
tn6530> z
%
```

You can also press CTRL/] and type the character z immediately at the TACL prompt.
To resume your TACL session from UNIX, you can use the cshell fg command. If more than one process (job) is suspended, you must specify which one you want to resume.

Stopping TN6530

Use the following procedure to stop the emulator utility and return to UNIX:

1. From the TACL prompt or other host system prompt, type logoff:

```
TACL 22> logoff
```

You should always log off the host system when you are through with a session, for security reasons.

2. Enter quit to stop the emulator:

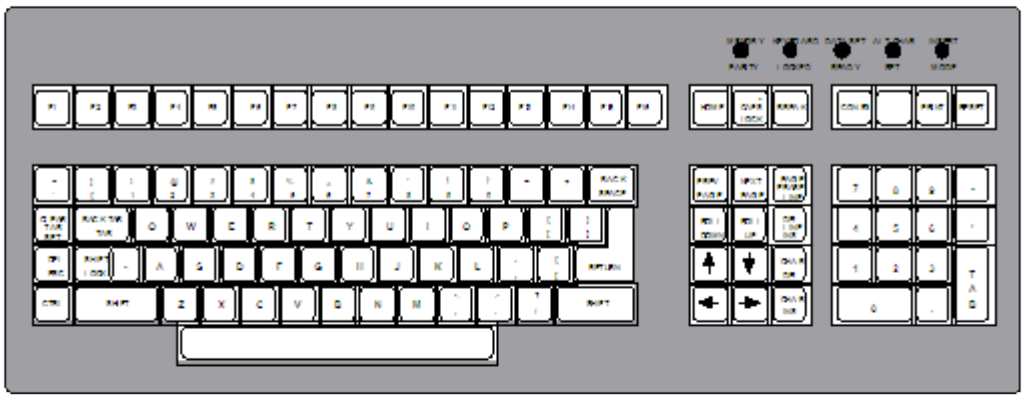
   ```
   tn6530> quit
   ```

   The UNIX prompt (%) appears again.

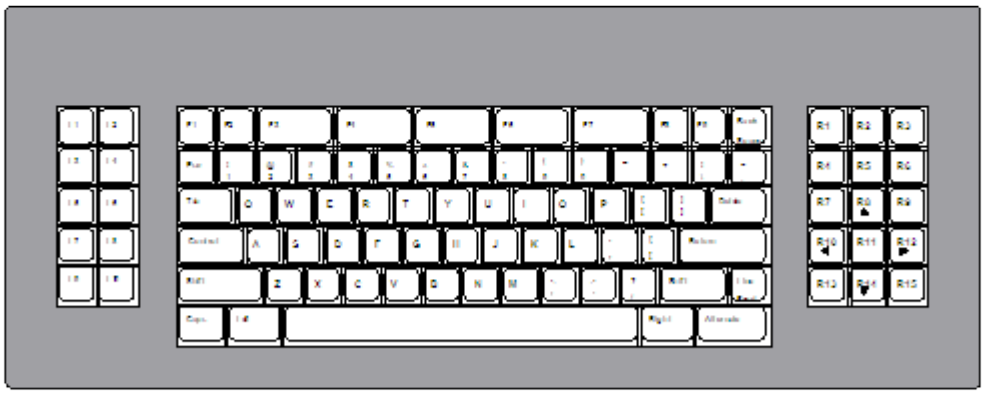
Keyboard Layouts

The 653X keyboard layout is different from the layout of your workstation. As a result, many functions are accessed differently on the two keyboards. Use the illustrations in Figure 3 to compare the 653X keyboard layout with a typical Sun workstation layout.

Figure 3 653X and Sun Keyboard Layouts



653X Keyboard



Sun Workstation
Keyboard

VST003.vsd

Cross-Reference Key Chart

Use Table 17 to find the Sun workstation keys that correspond to the 653X keys. This same information is contained in a keyboard mapping file named /etc/map6530. You can override this file by setting the environment variable MAP6530. For more information about keyboard mapping, see Appendix A, Keyboard Mapping for TN6530.

In the descriptions of the 653X keys, a slash (/) connects two keys that you press at the same time, and a hyphen (-) connects words that appear together on a key label.

In the descriptions of Sun workstation keys, a plus (+) indicates you press the keys at the same time, and LEFT/RIGHT indicates that you can press either the LEFT or RIGHT key.

Table 17 Sun Workstation and 653X Keys

Function	653X Keyboard	Sun Keyboard
Application-defined	F1	LEFT/RIGHT + 1
function keys	F2	LEFT/RIGHT + 2
	F3	LEFT/RIGHT + 3
	F4	LEFT/RIGHT + 4
	F5	LEFT/RIGHT + 5
	F6	LEFT/RIGHT + 6
	F7	LEFT/RIGHT + 7
	F8	LEFT/RIGHT + 8
	F9	LEFT/RIGHT + 9
	F10	LEFT/RIGHT + 0
	F11	LEFT/RIGHT + Q
	F12	LEFT/RIGHT + W
	F13	LEFT/RIGHT + E
	F14	LEFT/RIGHT + R
	F15	LEFT/RIGHT + T
	F16	LEFT/RIGHT + Y
Shifted function keys	SHIFT/F1	SHIFT + LEFT/RIGHT + 1
	SHIFT/F2	SHIFT + LEFT/RIGHT + 2
	SHIFT/F3	SHIFT + LEFT/RIGHT + 3
	SHIFT/F4	SHIFT + LEFT/RIGHT + 4
	SHIFT/F5	SHIFT + LEFT/RIGHT + 5
	SHIFT/F6	SHIFT + LEFT/RIGHT + 6
	SHIFT/F7	SHIFT + LEFT/RIGHT + 7
	SHIFT/F8	SHIFT + LEFT/RIGHT + 8
	SHIFT/F9	SHIFT + LEFT/RIGHT + 9
	SHIFT/F10	SHIFT + LEFT/RIGHT + 0
	SHIFT/F11	SHIFT + LEFT/RIGHT + Q
	SHIFT/F12	SHIFT + LEFT/RIGHT + W
	SHIFT/F13	SHIFT + LEFT/RIGHT + E
	SHIFT/F14	SHIFT + LEFT/RIGHT + R
	SHIFT/F15	SHIFT + LEFT/RIGHT + T
	SHIFT/F16	SHIFT + LEFT/RIGHT + Y
Cursor up	¦	R8
Cursor down	Ø	R14
Cursor right	Æ	R10
Cursor left	¨	R12

Table 17 Sun Workstation and 653X Keys *(continued)*

Function	653X Keyboard	Sun Keyboard
Cursor home	HOME	R7
Cursor home down	SHIFT/HOME	R11
Cursor to after last data	CTRL/HOME	R5
Cursor to beginning of line	SHIFT/RETURN	R3
Cursor to end of line	CTRL/RETURN	R6
Roll up one line	ROLL UP	R1
Roll down one line	ROLL DOWN	R4
Next page	NEXT PAGE	R15
Previous page	PREV PAGE	R9
Set tab	CLR-TAB-SET	F7
Clear tab	SHIFT/CLR-TAB-SET	F8
Clear all tabs	CTRL/SHIFT/CLR-TAB-SET	F9
Horizontal tab	BACK TAB-TAB	TAB
Back tab	SHIFT/BACK TAB-TAB	LEFT/RIGHT + TAB
Insert character	CHAR INS	LEFT/RIGHT + I
Insert line	DEL-LINE-INS	SHIFT + LEFT/RIGHT + I
Delete character	CHAR DEL	LEFT/RIGHT + O
Delete line	SHIFT/DEL-LINE-INS	SHIFT + LEFT/RIGHT + O
Insert mode	SHIFT/CHAR INS	R2
Erase to end of line	CTRL/PAGE-ERASE-LINE	LEFT/RIGHT + L
Erase to end of page	CTRL/SHIFT/PAGE-ERASE-LINE	SHIFT + LEFT/RIGHT + L
Break character	BREAK	CTRL + C
Soft reset	SHIFT/RESET	CTRL + LEFT/RIGHT +
Exit emulator		CTRL +] + Q
Suspend emulator		CTRL +] + z

You can perform a soft reset to clear all buffers and cancel any pending function key processing and I/O operations. Soft reset also turns off insert character mode, clears all error conditions and messages, and enables the keyboard. The action in response to pressing these keys depends on whether you are working in conversational mode or block mode. For more information, see "Working in Conversational Mode" (page 148) and ""Working in Block Mode" (page 148).

NOTE: Some Sun workstation applications may have assigned functions to the keys which may be different from the functions in the keyboard map file, or may have assigned different scan codes to specific keys. To use such applications, you may have to alter the entries in the map6530 file.

"Cross-Reference Key Chart" (page 143) lists 653X functions that are not supported by TN6530 (they do not have an equivalent workstation key).

Table 18 Unsupported 653X Functions

Function	653X Keystroke
Display configuration menu	CONFG
Return to normal display	SHIFT/CONFG
Display full configuration menu	CTRL/SHIFT/CONFG
Alternate character set	ALT/CHAR
Print page	PRINT
Modem disconnect	CTRL/SHIFT/BREAK
Hard reset	CTRL/SHIFT/RESET
Turn on 25th line status display	CTRL/NEXT PAGE
Turn off 25th line status display	CTRL/PREV PAGE
Execute self test	CTRL/SHIFT/0

Terminal Options and Attributes

TN6530 uses the TELNET protocol to communicate with the remote NonStop host system. While establishing the connection, the emulator and remote TELNET server process negotiate options using the TELNET protocol WILL, WONT, DO, and DONT commands. It negotiates the following options: suppress go ahead, binary, echo, and terminal type. All terminal type options are ignored except TELOPT_TTYPE, which is used as described in "Requesting Block Mode," in Section 12, Communicating With the TELNET Server.

Once the connection is established, the emulator enters full-duplex conversational mode with character-at-a-time input mode. The emulator enters block mode only at the request of the host processor. Some applications on the NonStop system will place your terminal in block mode.

The mode is displayed on the right side of the twenty-fifth line on your screen. TTY-FDX indicates full-duplex conversational mode and BLOCK indicates block mode. See "Working in Conversational Mode" and "Working in Block Mode", later in this section, for more information.

NOTE: You can create an initialization file to override some terminal options and attributes. See "Initializing Terminal Options" (page 147) later in this section.

Video Attributes

If your terminal is capable of supporting the following attributes, they can be used with TN6530:

- Normal video
- Reverse video
- Underline

TN6530 attempts to make use of all video attributes supported by the terminal, and it substitutes some other attribute for one that is not supported. For example, if blinking video is not supported, reverse video is used. If reverse video is not available, underline video is used. If underline is not available, then normal video is used. The following attributes are tried for dim video (half intensity) in the order indicated: underline video, reverse video, normal video.

Underline is not available on a standard Sun workstation.

When many blinking characters appear on the screen, the emulator appears to operate slowly because it is spending most of its time performing the blinking function. To avoid this apparent slowdown, you can disable the blinking function by putting the following line in your.tn6530init file:

```
blinking off
```

See “Initializing Terminal Options” (page 147).

40-Character Lines

TN6530 supports 40-character lines by assigning one column to each displayable character. Double-width characters are displayed as normal characters, and the right half of the screen is not used.

Data Attributes

TN6530 supports all data attributes that a 653X terminal supports, including the following attributes:

- Protected and unprotected fields
- Fields containing only alphabetic, numeric, or alphanumeric characters
- Fields containing only shifted characters
- Fields where auto-tab is turned off

Initializing Terminal Options

As system administrator, you can create a file named /etc/tn6530init to override the following terminal options:

- The number of pages of terminal memory
- The screen size
- The column in which the bell rings
- The attribute to be used when the blinking, dim, or underline attributes are not supported

Users can also create a file named $HOME/tn6530init in their home directory to override the settings of the file /etctn6530init. You do not need to create either file unless you want to override the default terminal options.

Both $HOME/tn6530init and /etc/tn6530init are text (ASCII) files containing one or more of the following lines:

```
pages  num-pages
screen screen-code
bellcolumn column-number
blinking { normal | reverse }
dim { normal | reverse }
underline { normal | reverse }
```

You can include a comment line in the file by entering a pound sign (#) as the first character, followed by the text of the comment.

“Initializing Terminal Options” (page 147) summarizes the values you can supply for each terminal option.

Table 19 Terminal Options

Option	Values
pages	*num-pages* can be any positive integer.The default number of pages is 10.
screen	*screen-code* can be: 0 for 80 characters by 24 lines 1 for 132 characters by 27 linesThe default screen code is 0.
bellcolumn	*column-number* can be an integer from 0 through 132.The default is 0, which means that the bell does not ring.
blinking	Normal specifies normal mode. Off turns off blinking.

Table 19 Terminal Options *(continued)*

Option	Values
dim	Reverse specifies reverse video.
underline	The default for each attribute is normal.

The following example illustrates a. tn6530init file. The screen option is set to 1 (132 characters by 27 lines), the bell will ring at column 72, and the underline is set to display as reverse video:

```
# terminal option settings (Jan 14, 1990)
screen   1
bellcolumn  72
underline reverse
```

Working in Conversational Mode

In conversational mode, all characters you enter at the keyboard are immediately sent to the remote host. The characters are not displayed on the screen until the remote host echoes them back. The only exceptions are a few control codes that affect the display but are not transmitted to the host.

Display memory is organized as one continuous page of 300 display lines, 24 of which appear at one time. You can scroll the displayed text up and down one line or one page at a time. Characters you enter or characters received from the host are displayed at the current cursor position. If you scroll, you may write over existing characters unless you return to the last line of text.

To speed up the display in conversational mode, line 25 is displayed on line 3 of the screen and may scroll off your screen.

Key Functions

In conversational mode, keys that invoke the following functions are ignored:

```
Cursor to beginning of line
Cursor to end of line
Back tab
Insert character
Insert line
Delete character
Delete line
Insert mode
```

Working in Block Mode

In block mode, the emulator transmits and receives data in blocks, and the host controls the format of each page and which page is displayed on the screen. You can type in several lines (up to a full screen) and edit the data before it is transmitted.

Block mode has two submodes: protect and nonprotect:

- Protect submode allows the host application to designate specific areas of the screen as protected; for example, a screen form might have protected and unprotected fields. The cursor skips over the protected areas.
- Nonprotect submode does not allow the designation of protected areas and is used by applications such as editing programs that allow you to determine the format of the data you enter. You can move the cursor to any position on the screen (except the status line) and enter any type of characters, depending on the data attributes the application defines for the field. The application can define an unprotected field with the following attributes:
    - Free entry—You can enter any type of character.
    - Alphabetic—You can enter only letters.

- Numeric—You can enter only numeric digits.
- Alphanumeric—You can enter only letters or digits.
- Full numeric—You can enter only numeric digits and characters used in the expression of numbers ($ + , - .).
- Full numeric with space—You can enter a space, as well as full numeric characters.
- Alphabetic with space-—You can enter a space, as well as alphanumeric characters.

If you enter invalid data, a message appears in the status line. A field may also be defined so that letters are shifted to uppercase.

Display memory is divided into ten logical pages, each consisting of 24 lines (1920 characters). Only one page is displayed at a time. The application program controls which page is currently displayed. Typically, you can request the application to display a new page by pressing a function key.

Each page has its own cursor. You cannot move the cursor off the displayed page.

Key Functions

In either type of block submode (protect or nonprotect), the following functions may be defined differently by the application:

```
Roll up on line
Roll down one line
Next page
Previous page
```

The application may also define the response to the ENTER key.

In block protect mode, you cannot set or clear tabs. Also, the keys that perform the following functions are ignored:

```
Insert line
Delete line
```

Local Editing Functions

In block mode, TN6530 supports all local editing functions of the 6530 terminal, including:

- Inserting and deleting single or multiple lines and single or multiple characters
- Positioning the cursor using the full set of cursor control keys
- Erasing the end of a line or page
- Moving the cursor to the beginning or end of a line or page

NOTE: You cannot use these functions unless you start the emulator in a window that is at least 80 columns by 25 rows. The emulator will not perform 6530 block-mode functions unless it is named TN6530.

14 Addressing Mail to SMTP Hosts

The SMTP Gateway

The Simple Mail Transfer Protocol (SMTP) gateway allows you to send and receive mail through an HP NonStop TRANSFER mail service (such as M6530 or PSMAIL) to and from users whose mail service is provided by an SMTP host on the internet. The SMTP gateway also acts as an intermediate node in relaying SMTP messages between SMTP hosts.

To use the SMTP gateway, you start your mail service in the usual way. If you have received any messages from an SMTP host, the messages will be in your inbox folder. To send a message, you use the same mail commands as usual, but you must address the message as described in this section.

See your mail service manual for specific instructions on sending and reading messages.

The SMTP Correspondent

The SMTP correspondent is a TRANSFER/MAIL collection and forwarding point to which you address all mail sent through the gateway. The SMTP correspondent is specified in the configuration file. Ask your system manager for the correspondent name on your system. See the *TCP/IP Configuration and Management Manual* for more information.

The examples in this manual address mail to an SMTP correspondent named SMTPGATE.

Addressing Mail

You can address a message to an SMTP host using one or the other of the following forms:

```
{ SMTP-correspondent  [@SMTP-host] ( recipient-address ) }
{                                                        }
{ user @host.domain                                      }


   recipient-address is:


      { internet-address }
      { usenet-path      }


   internet-address is:


      user @host.domain


   usenet-path is:


      [ host![host!]...]host!use
```

SMTP-correspondent
: is the SMTP gateway correspondent name through which you are sending the mail. SMTP gateway correspondent names are defined when an SMTP gateway is configured on a system.

See the See the *TCP/IP Configuration and Management Manual* or the *TCP/IP (Parallel Library) Configuration and Management Manual* for more information.

SMTP-host

is the name of the system on which the SMTP gateway correspondent is configured. You can send mail through an SMTP gateway on a system on the HP network. If the gateway correspondent is on your local system, you can omit this parameter.

recipient-address

is the address of the mail recipient. You can specify any address that conforms to RFC 822. The syntax description in the preceding box and the examples below show two of the basic ways to address messages, an internet address, or a usenet path. For additional forms of *recipient-address*, see RFC 822.

internet-address

is the internet address of the mail recipient.

usenet-path

is the usenet path to the mail recipient.

user

is the recipient's name as defined in the user's mail system.

A user name on a NonStop mail system typically has the form LAST_FIRST@NODE. For example, to address a message to James Tollison at a system named SYSABC of a company named Safe Bank, you use either of the following names:

```
TOLLISON_JAMES @SYSABC
TOLLISON_JAMES @SYSABC.SAFEBANK.COM.
```

host

is the host name of the recipient's system; in a usenet path, it is the name of a host along the path to the recipient's system.

domain

is the domain name of the recipient's system; for example, medlab.COM.

NOTE: NonStop systems cannot process UUCP mail, but they can forward this mail to a UUCP gateway operating on some other system on the network.

Examples

In all of the following examples, SMTPGATE is the name configured as the SMTP gateway correspondent on a system named tsys1.

In the following example, a message is addressed to a user named jenny on a host named medsys in domain Medlabs.COM:

```
To: SMTPGATE(jenny@medsys.Medlabs.COM)
```

In the next example, a user logged on to a NonStop system other than tsys1 sends a message to user jenny through the SMTP gateway on the tsys1 system:

```
To: SMTPGATE @tsys1(jenny@medsys.Medlabs.COM)
```

To address a message to a local recipient and to two recipients through the gateway, you enter a command such as the following one:

```
To: selby_amanda, SMTPGATE(jenny@medsys.Medlabs.COM),
  SMTPGATE(frank@datsys.Pharmco.COM)
```

In the following example, a message is addressed to user jerry on the medsys host by specifying a usenet path to the host:

```
To: SMTPGATE(abcsys!defsys!xyzsys!medsys!jerry)
```

15 Anonymous FTP

Anonymous FTP provides a means by which an archive site allows general access to the archives of information at that site. Each site creates a special “anonymous” logon, which has limited access rights to the archive hosts, as well as restrictions on operations. Generally, the only operations allowed are logging in to the site using FTP, listing the contents of a limited set of directories, and retrieving files.

Anonymous FTP

The HP NonStop FTP anonymous user support is primarily targeted for the Open System Services (OSS) environment. Although the FTP server can provide support for both the OSS and Guardian environments, users who configure their environment to support the Guardian anonymous user must assume responsibility for the security of their system due to the complexity associated with sites that fully utilize the NonStop Safeguard subsystem access control lists (ACLs).

The FTP server relies heavily on the security features of the underlying operating system. This includes the access permission setup for files and directories. Careful incorporation of these features is required of the system administrator in order to provide a secure site. It is necessary to have the Safeguard subsystem running and properly configured when supporting the NonStop FTP anonymous user.

The Safeguard subsystem is used to add an anonymous user and to create an authentication record that allows the user to log onto either the OSS or Guardian environment. Because the OSS environment coexists with the Guardian environment, the FTP server helps determine which initial file system or default personality the FTP anonymous user is permitted to logon to.

To logon to the OSS environment, the anonymous user enters the user name, “anonymous” or “ftp”. The FTP server checks this logon against the contents of the FTPUSERS file residing on `$SYSTEM.ZTCPIP`. If the logon is found in this file, the anonymous user logon is rejected. Refer to the subsection, Disallowing Logons in Section 7, Communicating With the FTP Server, for a description of how the FTPUSERS file is set up.

NOTE: For the OSS anonymous user who enters either “anonymous” or “ftp,”, you need add only the lowercase alias or aliases to the FTPUSERS file. The FTP server code converts the OSS anonymous logon to lowercase before checking it against the FTPUSERS file, since only the lowercase (OSS anonymous) alias is required to be added under SAFEGUARD.

For the OSS anonymous user, it is important that the system administrator correctly sets up the initial directory in order to allow the anonymous user to logon. The initial directory is what the FTP server uses for the new OSS root directory. It is also used by the server to determine the FTP default personality. Refer to the subsection, FTP Default Personality Selection in Section 6, FTP—Transferring Files, for an explanation of this topic. For an anonymous user, if an invalid or non-existent INITIAL-DIRECTORY is provided or if OSS is not running, the OSS anonymous user logon will be denied.

HP NonStop requires the OSS anonymous user aliases (“anonymous” and “ftp”) to be frozen under Safeguard for security purposes. The Guardian user name, `NULL.FTP` (for which “`anonymous`” and “`ftp`” are aliases), also should be frozen. This is to prevent the anonymous user logons from being accessed outside of FTP.

For security reasons, HP recommends that the objects (volumes/subvolumes, directory/subdirectories and files) which are accessible by the anonymous user not be owned by the anonymous user. Providing the anonymous user with a different user ID prevents the user from altering the security settings associated with these objects.

Users who must support anonymous FTP write operations must assume their own security risks associated with this capability. These users must also allow for their own disk resource management procedures. HP recommends against providing anonymous FTP write operations.

Make note of the following:

- The /E (Expand) or any directory on a remote system is not supported for access by the anonymous user.
- The /G (Guardian initial directory) is considered invalid for the anonymous user.
- The FTP command, `quote Guardian/OSS`, is disabled for the anonymous user.

The anonymous user can logon to the Guardian environment using the `NULL.FTP` user ID. For this to occur, the user ID must be thawed through Safeguard. This logon is also checked against the contents of the `FTPUSERS` file. System administrators allowing this capability should be versatile in configuring Safeguard ACLs. HP recommends that the `NULL.FTP` user ID remain frozen at all times. This capability (the NULL.FTP user ID) should be used only if all the proper steps have been taken to secure the system.

Checklist For A Secure FTP Anonymous Site

Note that the following checklist does not preclude the usage of additional information found in the Safeguard Administration Manual and Safeguard Reference Manual.

- The anonymous user `NULL.FTP` should be configured in the Safeguard database as shown:

  ```
  SAFECOM
  =add user NULL.FTP, 0,15, owner 255,255, PASSWORD guest, &
  Guardian security "OOOO", Guardian volume <$vol.subvol>
   =info user NULL.FTP, detail.
  ```

  The password "`guest`" is entered in lowercase. The password attribute is not required if this account is to be frozen and not used (recommended by HP).

  The user ID of `NULL.FTP` must designate a `<group-num>` of zero and the `<user-num>` must be a number from 1 to 255. Zero should not be used for the `<user-num>`.

  The Guardian default disk file security is restricted to owner access only.

  The Guardian volume is the name of the default volume and subvolume (that is, `$guest.ftp`) specifically setup by the system administrator for the `NULL.FTP` anonymous user.

  The info command is used to display the attributes stored in the user's authentication record.

- After the Guardian anonymous user `NULL.FTP` is added, the OSS anonymous user can be configured using the `ALIAS` command, as shown:

  ```
  SAFECOM
  =add alias anonymous, NULL.FTP, PASSWORD guest, &
  INTIAL-DIRECTORY <dir-path>
  =info alias anonymous, detail
  =add alias ftp, NULL.FTP, PASSWORD guest, &
  INITIAL-DIRECTORY <dir-path>
  =info alias ftp, detail
  ```

  Note that the "anonymous" and "ftp" aliases are only required to be entered in lowercase. The FTP server is case insensitive when checking for the OSS anonymous user logon.

  The password "guest" is entered in lowercase.

  The INITIAL-DIRECTORY is the directory pathname (that is, /guest/ftp) specifically setup by the system administrator for the anonymous FTP user.

  The INFO command is used to display the attributes stored in the alias authentication record.

- To disable the anonymous user support (recommended by HP) in the Guardian environment, the system administrator should freeze the Guardian `NULL.FTP` user as shown:

  ```
  SAFECOM
  =freeze user NULL.FTP
  ```

NOTE: If Safeguard is brought down for any reason, although the frozen users remain frozen, they behave as if they were unfrozen (or thawed) thus losing the added security provided.

To enable the Guardian anonymous user support, the `NULL.FTP` user needs to be thawed. However, once this is done, the `NULL.FTP` logon with the password "guest" becomes accessible outside of FTP. Sites using Safeguard ACLs to secure their system, must take this into consideration and secure not only volumes, subvolumes and files, but also processes, terminals and devices.

- For a secure OSS environment, the system administrator should freeze the alias "anonymous" and "ftp" user (for the purpose of disabling the anonymous access outside of FTP) as shown:

```
SAFECOM
=freeze alias anonymous
=freeze alias ftp
```

- In the OSS environment, the initial root directory (that is, `/guest/ftp`) and other subdirectories specifically set up by the system administrator for the anonymous user should be owned by the super user ID or some special user ID. By not allowing these directories to be owned or reside in the same group as the anonymous user prevents the anonymous user from altering the security access. The OSS initial directory should not be a `/E` or any directory on a remote system because this is invalid and the anonymous logon will be denied. The `/G` (Guardian initial directory) is also invalid and will cause the anonymous logon to be denied.

△ **CAUTION:** Users who wish to support the anonymous FTP writes must be aware of the inherent security risks. In order to provide this capability in a secure manner, one currently known safe practice is to put all the files generated by anonymous FTP writes into a write-only area (that is, not readable by the anonymous FTP user). After careful inspections and approvals have been made according to site policies, these files are later manually moved into a read-only area (for the anonymous FTP user) by the FTP system administrator.

- Access to the OSS files and directories is controlled by the POSIX.1 file permission bits. No additional level of security is provided. Using the `CHMOD` command, the nine permission bits should be setup to restrict the anonymous user's files and directories access. The nine permission bits can be shown with the list long directory contents command as follows:

```
/guest/ftp: ls -l

total 987
drwxr-xr-x 1 super super  1024 Nov 22 15:58 tprog
-rw-r--r-- 1 super super 36001 Mar 7 10:47 tcp1_rfc
-rw-r--r-- 1 super super 34000 Mar 8 11:00 tcp2_rfc
-rw-r--r-- 1 super super 27500 Aug 20 13:00 tcp3_rfc
-rw-r----- 1 super super 10370 Jun 27 11:00 xfile
```

The directory (d) or file (-) field precedes the three types of user (Owner, Group, Other) with three types of permissions (read, write, execute).

The anonymous user would be categorized as an "Other" user. Therefore, the last three permission bits determine the anonymous user access. The `r` (read) and `x` (execute) permission bits on a directory must be set to allow "Other" users (that is, FTP anonymous users) to access the directory.

- In the OSS environment, care should be taken that no OSS files under the anonymous user root directory are linked to Guardian files. In other words, symbolic links connecting OSS files to `/G` Guardian files must not be configured for the OSS anonymous user. Users should not have any symbolic links referring to directories or files outside the subtree, since such directories and/or files will be inaccessible to the anonymous user. It is best to avoid all absolute symbolic links (those whose contents begin with "/") within the subtree, since these links will be

interpreted relative to the root of the subtree (that is, the INITIAL-DIRECTORY) rather than relative to the root directory of the root OSS fileset.

- In regard to the usage of NFS mounted files, the system administrator should understand how the security on these files differs from the standard files. Anonymous FTP has no mechanism to differentiate these files from the standard files. Based on how NFS files are mounted and setup, the POSIX.1 permission bits may not necessarily reflect its access capability. Also if the anonymous user is allowed to own NFS files, all the NFS access capabilities open up to this user.

A Keyboard Mapping for TN6530

Introduction

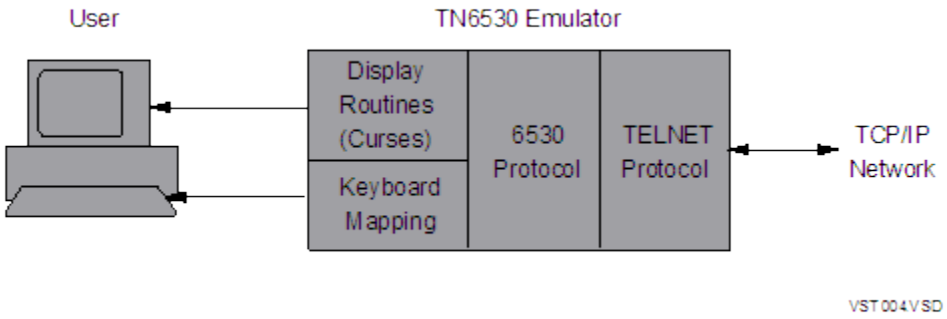
The TN6530 is a multiple-page terminal emulation utility that runs on a BSD-based UNIX system. The version of the utility described here runs on Sun Microsystems machines; however, the emulator is not limited to Sun machines. The TN6530 code is fully portable to any BSD-based UNIX system.

The main objective of this utility is to provide users on UNIX systems the capability to log on to a NonStop system and run applications that depend on the capabilities of a 6530 terminal or a similar multiple-page terminal. TN6530 sends the keystrokes you enter from your workstation keyboard to the host. The host application program you are using processes the data. The application may also send information to the workstation; this information appears on your screen along with the characters you enter.

The following major functional units of TN6530 are shown in Figure 4:

- TELNET protocol
- 6530 protocol
- keyboard mapping
- display routines

Figure 4 TN6530 Functional Units



TELNET Protocol

Access to the Nonstop system is provided through the TELNET protocol running as a client on a TCP/IP network using the sockets library. From the user's perspective, TN6530 can be considered a frontend to the TELNET client. TN6530 supports all commands and options of the 4.3 BSD TELNET client. These commands and options are described in Section 11, TELNET—Using a Network Virtual Terminal

Display Routines

Screen manipulation and display is achieved by the use of the `curses` routines. These routines provide a means of writing portable terminal-dependent code for screen manipulation, such as cursor movement and editing.

Supported and Unsupported Features

Supported Features

- Conversational and Block mode
- Duplex and half duplex in Conversational mode, Keyboard-generated soft reset
- Keyboard-generated soft reset

- Inserting/deleting single or multiple lines
- Cursor control keys for easy cursor positioning
- Erasing end of line and erasing end of page
- Data organization and protection through:
  - Protected and unprotected fields
  - Fields containing only alphabetic, numeric, alphanumeric characters
  - Fields containing only shifted characters
  - Fields where auto-tab is disabled

Unsupported Features

- User-selectable cursor type (block or underline, blinking or non-blinking)
- Control-character display mode
- Diagnostic programs
- Terminal options (for example, printer option, bar code reader, magnetic stripe reader, and so forth)
- Alternate character set
- Graphics character set

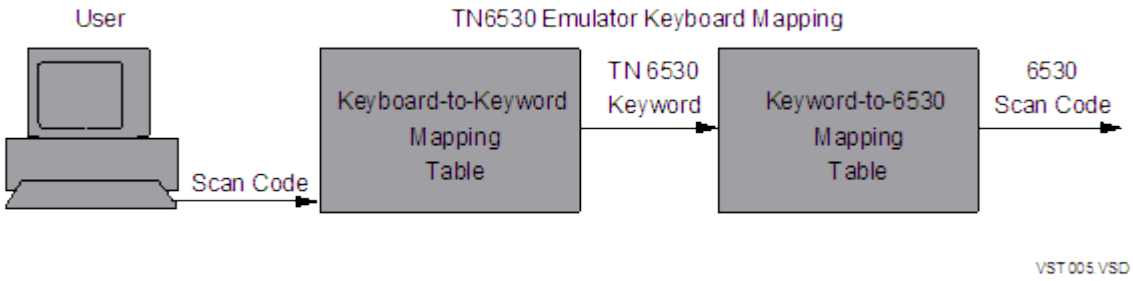
Foreign Language Support

The local computer system, and the local terminal on which the program runs, must provide support for languages other than English, if desired.

Keyboard Mapping

This subsection describes keyboard mapping for TN6530, which is designed to operate on a variety of terminals from various manufacturers. Each of these terminals can have different keyboards that generate different scan codes. TN6530 operates with different keyboards by mapping the keyboard's scan codes to a set of standard keywords, as shown in "Keyboard Mapping" (page 157). TN6530 interprets the keywords as equivalent 653X functions and sends the appropriate 653X scan codes to the host.

Figure 5 TN6530 Keyboard Mapping



The mapping of a particular terminal's scan codes to the set of TN6530 keywords listed in Table 20 is accomplished by individual keyboard mapping tables. Before starting TN6530, you specify the type of terminal you are using through the UNIX environment variable TERM.

The 653X functions (keystrokes) listed in Table 20 are supported by TN6530. The single set of standard keywords allows any keyboard to be mapped to perform the same functions as a 653X terminal. Table 22 lists the keystroke(s) to TN6530 keyword mapping for a Sun keyboard. This mapping is contained in the MAP6530 file that is copied when installing TN6530 (see Appendix C,

Installing TN6530 on a Sun Workstationfor more information on this file). The key chart presented in Section 10 was compiled from Table 20 and Table 22.

Table 20 Mapping Table—TN6530 Keyword-to-653X Function

TN6530 Keyboard	653X Keystrokes	Functional Description
f1	F1	Unshifted F1
.	.	.
.	.	.
f16	F16	Unshifted F16
shift_f1	SHIFT/F1	Shifted F1
.	.	.
.	.	.
shift_f16	SHIFT/F16	Shifted F16
rollup	ROLL UP	Roll up
srollup	SHIFT/ROLL UP	Roll up (*)
rolldown	ROLL DOWN	Roll down
srolldown	SHIFT/ROLL DOWN	Roll down (*)
nextpage	NEXT PAGE	Next page
snextpage	SHIFT/NEXT PAGE	Next page (*)
prevpage	PREV PAGE	Prev page
sprevpage	SHIFT/PREV PAGE	Prev page (*)
tabset	TAB SET	Set tab
tabclr	SHIFT/CLR TAB	Clear tab
tabclrall	CTRL/SHIFT/CLR TAB	Clear all tabs
tab	TAB	Horizontal tab
backtab	SHIFT/BACK TAB	Back tab
insline	INS LINE	Insert line
delline	SHIFT/DEL LINE	Delete line
inschar	CHAR INS	Insert character
delchar	DEL CHAR	Delete character
insert	SHIFT/CHAR INS	Insert mode
eeol	CTRL/ERASE LINE	Erase to end of line
eeop	CTRL/SHIFT/ERASE PAGE	Erase to end of page up
up	¦	Cursor up
down	Ø	Cursor down
left	¨	Cursor left
right	Æ	Cursor right
home	HOME	Cursor home
homedown	SHIFT/HOME	Cursor home down
last	CTRL/HOME	Cursor to after last data

Table 20 Mapping Table—TN6530 Keyword-to-653X Function *(continued)*

TN6530 Keyboard	653X Keystrokes	Functional Description
begin	SHIFT/RETURN	Cursor to beginning of line
end	CTRL/RETURN	Cursor to end of data
break	BREAK	Break character
reset	SHIFT/RESET	Soft reset
escape	CTRL/]	TELNET escape
(*) Function of key sequence is application-dependent.		

Unsupported Functions

The 653X functions (keystrokes) listed in "Unsupported Functions" (page 159) are not supported by TN6530 and therefore do not have an equivalent emulator keyword.

Table 21 Unsupported 653X Functions

653X Keystroke(s)	Functional Description
CONFG	Display configuration menu
SHIFT/CONFG	Return to normal display
CTRL/SHIFT/CONFG	Display full configuration menu
ALT CHAR	Alternate character set
PRINT	Print page
CTRL/SHIFT/BREAK	Modem disconnect
CTRL/SHIFT/RESET	Hard reset
CTRL/NEXT PAGE	Turn on 25th line status display
CTRL/PREV PAGE	Turn off 25th line status display
CTRL/SHIFT/0	Execute self test

Reading TN6530 Keyboard Mapping Tables

Each entry in a keyboard's mapping table consists of one of the keywords listed in Table 20, followed by an equal character (=), followed by the scan code sequence (enclosed in quotes) that is to be mapped to that keyword. The scan code may be an octal value, a control sequence, or an escape sequence. The scan code may be followed by a vertical bar character (|) and another scan code, allowing a specific 653X function to be performed by more than one keystroke or set of keystrokes. For example, you could specify the left arrow function to be performed by either the left arrow key or the backspace key, because the backspace key on a 653X terminal is nondestructive.

Control sequences are specified by preceding the character(s) inside the single quotes with a circumflex (^); for example, CTRL/A is specified as '^a'. An escape sequence is specified by preceding the character(s) inside the single quotes with a backslash (\), followed by an uppercase letter E; for example, ESC/A is specified as '\Ea'. Any character can be specified by entering a backslash (\) followed by its octal value. This feature is especially useful when specifying keystrokes that generate scan codes in the range%200 through%377 (the extended ASCII characters, 128 through 255 decimal). On many terminals scan codes in this range do not produce displayable characters.

A pound sign character (#) anywhere in the file indicates a comment. Everything from the pound sign character to the end of the line is ignored.

The mapping tables provided contain maps for all currently supported terminal types. Table 22 is the keyboard mapping table for a Sun Microsystems keyboard.

Table 22 Mapping Table—Sun Keyboard to TN6530 Keyword

Emulator Keyword	=Sun Scan Code;	Sun Keystroke(s)
f1 =	'\261';	# LEFT/RIGHT - 1
f2 =	'\262';	# LEFT/RIGHT - 2
f3 =	'\263';	# LEFT/RIGHT - 3
f4 =	'\264';	# LEFT/RIGHT - 4
f5 =	'\265';	# LEFT/RIGHT - 5
f6 =	'\266';	# LEFT/RIGHT - 6
f7 =	'\267';	# LEFT/RIGHT - 7
f8 =	'\270';	# LEFT/RIGHT - 8
f9 =	'\271';	# LEFT/RIGHT - 9
f10 =	'\260';	# LEFT/RIGHT - 0
f11 =	'\361';	# LEFT/RIGHT - Q
f12 =	'\367';	# LEFT/RIGHT - W
f13 =	'\345';	# LEFT/RIGHT - E
f14 =	'\362';	# LEFT/RIGHT - R
f15 =	'\364';	# LEFT/RIGHT - T
f16 =	'\371';	# LEFT/RIGHT - Y
shift_f1 =	'\241';	# SHIFT- LEFT/RIGHT - 1
shift_f2 =	'\300';	# SHIFT- LEFT/RIGHT - 2
shift_f3 =	'\243';	# SHIFT- LEFT/RIGHT - 3
shift_f4 =	'\244';	# SHIFT- LEFT/RIGHT - 4
shift_f5 =	'\245';	# SHIFT- LEFT/RIGHT - 5
shift_f6 =	'\336';	# SHIFT- LEFT/RIGHT - 6
shift_f7 =	'\246';	# SHIFT- LEFT/RIGHT - 7
shift_f8 =	'\252';	# SHIFT- LEFT/RIGHT - 8
shift_f9 =	'\250';	# SHIFT- LEFT/RIGHT - 9
shift_f10 =	'\251';	# SHIFT- LEFT/RIGHT - 0
shift_f11 =	'\321';	# SHIFT- LEFT/RIGHT - Q
shift_f12 =	'\327';	# SHIFT- LEFT/RIGHT - W
shift_f13 =	'\305';	# SHIFT- LEFT/RIGHT - E
shift_f14 =	'\322';	# SHIFT- LEFT/RIGHT - R
shift_f15 =	'\324';	# SHIFT- LEFT/RIGHT - T
shift_f16 =	'\331';	# SHIFT- LEFT/RIGHT - Y
up =	'\E[A';	# R8
down =	'\E[B';	# R14
right =	'\E[C';	# R10

Table 22 Mapping Table—Sun Keyboard to TN6530 Keyword *(continued)*

Emulator Keyword	=Sun Scan Code;	Sun Keystroke(s)
left =	'\E[D';	# R12
home =	'\E[214z';	# R7
homedown =	'\E[218z';	# R11
rollup =	'\E[208z';	# R1
srollup =	'\E[228z';	# F5
rolldown =	'\E[211z';	# R4
srolldown =	'\E[229z';	# F6
nextpage =	'\E[222z';	# R15
snextpage =	'\E[227z';	# F3
prevpage =	'\E[216z';	# R9
sprevpage =	'\E[226z';	# F4
insert =	'\E[209z';	# R2
last =	'\E[212z';	# R5
begin =	'\E[210z';	# R3
end =	'\E[213z' | '\E[220z';	# R6 | R13
tabset =	'\E[230z';	# F7
tabclr =	'\E[231z';	# F8
tabclrall =	'\E[232z';	# F9
tab =	'^I';	# TAB
backtab =	'\211';	# LEFT/RIGHT - TAB
inschar =	'\351';	# LEFT/RIGHT - I
insline =	'\311';	# SHIFT- LEFT/RIGHT - I
delchar =	'\344';	# LEFT/RIGHT - O
delline =	'\304';	# SHIFT- LEFT/RIGHT - O
eeol =	'\354';	# LEFT/RIGHT - L
eeop =	'\314';	# SHIFT- LEFT/RIGHT - L
break =	'^C';	# Control - C
reset =	'\236';	# Control - LEFT/RIGHT - `
escape =	'^]';	# Control -]

B Error Messages

This appendix explains the cause and effect of messages produced by the TCP/IP application client programs and TN6530. The message descriptions include suggestions on how to recover from an error. The messages for each application are presented separately and arranged alphabetically. See the manual describing the mail service you are using for information about errors that appear when you are sending mail.

Recovering From Errors

You can apply the following general approach to recover from errors you encounter while using the TCP/IP client programs:

- Make sure the command you have entered has valid parameters, such as the host name or address, the user name or ID, or the port number. Some errors result from a simple typing mistake.

  If you have made a typing error, try the command again. If the error is more complicated, you can ask your system manager for help in locating the problem. If necessary, you can also contact the system administrator of the remote system.

- Communication problems that result in socket error messages can occur. These error messages include an *error-reason* that is returned from a TCP/IP sockets library routine named error or from the Guardian file system. To find a full explanation of these errors, refer to the *TCP/IP Configuration and Management Manual, the TCP/IP (Parallel Library) Configuration and Management Manual, or the TCP/IPv6 Configuration and Management Manual*, which describe the *error-reason* values in an appendix. The descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual* also explain some *error-reason* values you might encounter, as well as the *error-number* values included in some messages.
- In some cases, problems occur because the TCP/IP process is no longer running. Ask your system manager to check whether the TCP/IP process has stopped.

ECHO Error Messages

This subsection describes the ECHO error messages.

`Connection Refused`

Cause

The remote system is accessible, but it has no active ECHO server.

Effect

Your request for an ECHO connection terminates.

Recovery

Contact the system administrator of the remote system to determine whether an ECHO server should be available.

`connection timeout`

Cause

No reply has been received from the remote system within the TCP/IP system-defined timeout interval.

Effect

Your ECHO connection terminates.

Recovery

Contact the system administrator of the remote system to determine why the ECHO server is not responding, or try again later.

`Name does not resolve to supplied parameters.`

Cause

You have specified a host name or address in the ECHO run command that is not known to the system, or the ECHO client cannot locate a HOSTS file on your system.

Effect

Your request for an ECHO connection terminates.

Recovery

Enter the ADD DEFINE =tcpip^host^FILE command with the appropriate file name before you enter the ECHO run command. If you do not know the Guardian name of the HOSTS file, ask your system manager where the file is located.

`network is unreachable`

Cause

At this time, ECHO cannot establish a connection with the network you specified in a host name or host address.

Effect

Your request for an ECHO connection terminates.

Recovery

Try to connect with the network later.

FINGER Error Messages

This subsection describes the FINGER error messages.

`connect: error-reason`

Cause

FINGER could not establish a connection to the host; *`error-reason`* is a socket error.

Effect

The information that you are requesting cannot be received by your local system.

Recovery

See "Recovering From Errors" (page 162).

`finger: Invalid user name`

Cause

The user name you specified is not valid on the remote host.

Effect

The FINGER operation terminates.

Recovery

Determine the correct user name and try to run FINGER again.

`finger: unknown service`

Cause

The entry for FINGER is missing from the $SYSTEM.ZTCPIP.SERVICES file, so the system could not determine which TCP port to use.

Effect

The FINGER operation terminates.

Recovery

Ask the system manager or operator to place an entry for FINGER in the SERVICES file.

`Name does not resolve to supplied parameters.`

Cause

The internet address for the host name you specified cannot be determined.

Effect

The FINGER operation terminates.

Recovery

Make sure you have entered the correct host name. If the name is correct, ask the system manager or operator to check whether the host name is defined with its internet address in the HOSTS file on your system.

`socket: error-reason`

Cause

FINGER could not create a socket for the reason indicated by *error-reason*.

Effect

The FINGER operation terminates.

Recovery

See "Recovering From Errors" (page 162).

FTP Error Messages

This subsection describes the FTP error messages.

FTP Client Error Messages

These error messages appear on the client's screen.

`abort: error-reason`

Cause

An error occurred while the system was trying to abort the connection.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`command send: error-reason`

Cause

The system failed to send data on a socket.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`ftp: bind: error-reason`

Cause

An error occurred when the system tried to associate a socket with a specific local internet address and port number.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`ftp: Can't convert filename to internal name`

Cause

The command you entered caused FTP to try to create a local file name by default from the file name you specified. The name cannot be converted.

Effect

The command operation fails.

Recovery

Enter the command again and specify both file names.

`ftp: connect` *`error-reason`*

Cause

FTP could not establish a connection for the reason indicated by *`error-reason`*. The network might be unreachable at this time, or the remote host might have rejected the connection.

Effect

The connection is not established.

Recovery

See "Recovering From Errors" (page 162).

`ftp: connect to address` *`address`*`:` *`error-reason`*

Cause

FTP failed to establish a connection to the specified address for the reason indicated by *`error-reason`*.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`ftp: ftp/tcp unknown service`

Cause

The entry for FTP is missing from the $SYSTEM.ZTCPIP.SERVICES file, so FTP cannot determine which TCP port to use.

Effect

The FTP command fails.

Recovery

Ask the system manager or operator to place an entry for FTP in the SERVICES file.

`ftp: getsockname` *`error-reason`*

Cause

FTP could not determine the data socket address.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`ftp: setsockopt` *`error-reason`*

Cause

FTP could not set the socket options. The TCP/IP protocol might not know a specific option.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`ftp: setsockopt (ignored)` *`error-reason`*

Cause

An error occurred when FTP tried to set the socket options.

Effect

The error is ignored and the operation continues.

Recovery

See "Recovering From Errors" (page 162).

`ftp: setsockopt SO_LINGER : error-reason`

Cause

FTP could not set the socket option SO_LINGER. This option causes the system to wait for data transfer to complete before closing the connection.

Effect

Connections might be closed before data transfer completes.

Recovery

See "Recovering From Errors" (page 162).

`ftp: socket: error-reason`

Cause

FTP could not create a socket for the reason indicated by *error-reason*.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

local-filename `:` *error-reason*

Cause

FTP had problems operating on the specified local file for the reason indicated by *error-reason*.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`Login failed`

Cause

FTP could not establish a session with the remote system because the user name or password you specified is not valid.

Effect

You are not logged on to the remote system.

Recovery

Try the command again with the correct user name and password.

`Mismatch! (Old filecode` *file-code*`) Delete file first!`

Cause

You tried to retrieve a file to replace an existing file, but the file codes of the two files are not the same.

Effect

The retrieval operation fails.

Recovery

Delete the old file and enter the command again to retrieve the new file.

`netin: error-reason`

Cause

The specified error occurred while FTP was receiving data from the network. The socket might have shut down, the connection might have timed out, or the connection might have been reset at the remote host.

Effect

The command fails. The data you requested might be incomplete.

Recovery

See "Recovering From Errors" (page 162).

`netout: error-reason`

Cause

The specified error occurred while FTP was sending data to the network. The socket might have shut down, the connection might have timed out, or the connection might have been reset at the remote host.

Effect

The operation fails. The remote system does not receive all the data.

Recovery

See "Recovering From Errors" (page 162).

`No control connection for command error-reason`

Cause

The FTP command/control connection was dropped for the reason specified by *error-reason*.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162)ix.

`Out of memory`

Cause

FTP could not allocate enough memory to expand the wild-card name you specified.

Effect

The operation fails.

Recovery

Specify names explicitly, or try the command again with the wild-card name.

`Unknown ftpcstm keyword key-word`

Cause

The FTPCSTM file contains an incorrect entry.

Effect

In the FTPCSTM file, commands that follow the incorrect entry are not executed.

Recovery

Exit FTP and edit the FTPCSTM file to correct the entry. The acceptable keywords are machine, login, password, account, macdef, and any FTP command.

FTP API Error Messages

These error messages are returned to the application.

```
400 Command Invalid
```

Cause

The FTP Server does not recognize the command.

Effect

The FTP Sever does not execute the command.

Recovery

Determine whether the command you issued is valid.

```
401 Not connected to the server
```

Cause

The application is not connected to the server.

Effect

FTP does not execute the command.

Recovery

The application must connect to the server before issuing the command.

```
402 Command issued failed
```

Cause

The command failed.

Effect

The command is returned with error messages.

Recovery

Check the error messages and take appropriate action. Determine whether the command you issued is valid.

```
403 Return text truncated
```

Cause

The buffer specified was too small to hold the text that was returned.

Effect

The buffer contains only a portion of the text that was returned

Recovery

Increase the size of the buffer.

```
404 Unknown option with the command
```

Cause

The options specified with the command were invalid.

Effect

The command is not executed,

Recovery

Specify valid options and reissue the command. See Section 8, FTP API External Specification for valid options.

```
405 Fail to create an FTP session
```

Cause

The application program using the FTP API failed to create the FTP client process.

Effect

The FTP session is not created. All subsequent FTP API calls fail.

Recovery

Determine whether the FTPGM points to a valid FTP object; that is, determine whether

- The object is present at the given location,
- The object has a proper file code (100)
- The user under whose ID the process is running has execute permission for the object file

Correct any error, and restart the application.

`406 Fail to open communication with FTP client`

Cause

The application program using the FTP API cannot communicate with the FTP Client Process.

Effect

All subsequent FTP API calls fail.

Recovery

Restart the application.

`407 Fail to send start up message to FTP client`

Cause

The application using the FTP API is unable to contact the FTP Client Process.

Effect

All subsequent FTP API fail.

Recovery

Restart the application.

`408 Wrong version of FTP client. Does not match with this API`

Cause

The application using the FTP API attempted to use the wrong version of the FTP Client.

Effect

The FTP client process is stopped and all subsequent API calls using that handle fail, returning the error ILLEGAL_HANDLE.

Recovery

Make sure that the FTPPGM point to the correct version of FTP. Then, restart the application.

`409 FTP handle used is invalid`

Cause

The FTP connection table does not contain the handle.

Effect

The current API call and all subsequent calls fail.

Recovery

Restart the application.

`411 Already connected`

Cause

The application program using the FTP API is already connected.

Effect

The new connection request is not processed.

Recovery

Close the existing connection and attempt to connect again.

`412 File not found`

Cause

The file is not present in the specified location.

Effect

The command issued by the application fails.

Recovery

Reissue the command, specifying the proper location of the file.

```
413 Invalid Usage
```

Cause

The usage is wrong.

Effect

The command is not executed.

Recovery

Examine the usage in FTPexpth, correct the call, and restart the application.

```
414 Command invalid on proxy connection
```

Cause

The command is applicable only to direct connections.

Effect

The command fails to execute.

Recovery

Issue the command only for direct connections.

```
415 Filename invalid
```

Cause

The file name is invalid.

Effect

The command fails to execute.

Recovery

Specify the file name in the proper format and restart the application.

```
416 Login failed
```

Cause

The login using the given name and password failed.

Effect

The application is connected to the server, but is not logged in. Remote commands are not executed.

Recovery

Log in using a valid user name and password.

```
419 Failed to open stderr
```

Cause

STDERR could not be opened.

Effect

The FTP session is not created.

Recovery

Specify a valid STDERR when you start the application.

```
423 Invalid parameters
```

Cause

One or more of the command parameters were invalid.

Effect

The command fails to execute.

Recovery

Specify valid parameter and restart the application. See Section 8, FTP API External Specification for valid parameters.

`424 Unable to get startup message`

Cause

The application program using the FTP API failed to get the startup message to send to the FTP Client Process.

Effect

The FTP session is not created.

Recovery

Restart the application

TELNET Error Messages

This subsection describes the TELNET error messages.

`Ambiguous mode` *`mode`* `('mode ?' for help)`

Cause

The mode you specified does not contain enough characters to identify it clearly.

Effect

The command is ignored.

Recovery

Enter the command again, but specify enough characters to identify it clearly as either character mode or line mode.

`Can't open terminal`

Cause

TELNET could not open your home terminal on the remote system.

Effect

You are not connected to the remote system.

Recovery

Make sure you have specified the host name or address correctly. If the TELNET port is not 23, specify the port number.

`Can't setmode` *`number`* `on term err error-number`

Cause

TELNET cannot set the mode you specified due to the error specified by *`error-number`*.

Effect

The mode is not set.

Recovery

See "Recovering From Errors" (page 162).

`Can't turn off/on crmod err` *`error-number`*

Cause

An error occurred when TELNET tried to switch modes from line-to-character or character-to-line.

Effect

The mode is not switched.

Recovery

See "Recovering From Errors" (page 162).

`Can't turn off/on echo err error-number`

Cause

An error occurred when TELNET tried to turn echo on or off.

Effect

Echoing remains in the same state as it was before you issued the command.

Recovery

See "Recovering From Errors" (page 162).

`Can't turn off/on spacing err error-number`

Cause

An error occurred when TELNET tried to turn carriage return mode on or off.

Effect

The spacing is not changed.

Recovery

See "Recovering From Errors" (page 162)x.

`Connection closed by foreign host`

Cause

The remote host closed your TELNET connection. This message appears when you close the connection.

Effect

Your session with the remote host is terminated.

Recovery

If you did not terminate the session, try again later.

`Open of my-term error error-number`

Cause

A Guardian file-system error occurred when TELNET tried to open the virtual terminal file process.

Effect

The connection you requested is not made.

Recovery

See "Recovering From Errors" (page 162).

`READX error (error-number) on term fd file-number`

Cause

A read error occurred when TELNET tried to read the specified file.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

`Service not available`

Cause

The service you selected is not available through the remote TELNET server.

Effect

You cannot use the service at this time.

Recovery

Tell the system manager of the remote system that the service has not been started. Try again after the service is started.

`setsockopt can't wait: error-reason`

Cause

A communications problem has occurred. The TCP/IP process might not be running; for example, an operator might have stopped it.

Effect

TELNET terminates.

Recovery

Ask your system manager if the TCP/IP process is running. See "Recovering From Errors" (page 162).

`setsockopt failure: error-reason`

Cause

A communications problem has occurred. The TCP/IP process might not be running; for example, an operator might have stopped it.

Effect

TELNET terminates.

Recovery

Ask your system manager if the TCP/IP process is running. See "Recovering From Errors" (page 162).

`setsockopt SO_DEBUG: error-reason`

Cause

A communications problem has occurred. The TCP/IP process might not be running; for example, an operator might have stopped it.

Effect

TELNET terminates.

Recovery

Ask your system manager if the TCP/IP process is running. See "Recovering From Errors" (page 162).

`Telnet: connect error-reason`

Cause

TELNET was unable to connect to the host for the reason specified by *error-reason*.

Effect

The connection you requested is not established.

Recovery

See "Recovering From Errors" (page 162).

`Telnet 'send' error - argument disappeared!`

Cause

TELNET detected an internal error while parsing the send command you entered.

Effect

The command is not executed.

Recovery

Try entering the command again. Make sure that you are following the required syntax.

`Telnet: tcp/telnet unknown service`

Cause

The entry for TELNET is missing from the $SYSTEM.ZTCPIP.SERVICES file, so TELNET cannot determine which TCP port to use.

Effect

The command fails.

Recovery

Ask the system manager or operator to place an entry for TELNET in the SERVICES file. The entry should specify TCP port 23.

`toggle: ambiguous argument ('toggle ?' for help)`

Cause

The argument you specified in a toggle command does not contain enough characters to identify it clearly.

Effect

The command is ignored.

Recovery

Enter the command again, but specify enough characters to identify it clearly as one of the following arguments: autoflush, autosynch, crmod, debug, localchars, netdata, or options.

`toggle: unknown argument ('toggle ?' for help)`

Cause

The argument you specified in a toggle command is not valid.

Effect

The command is ignored.

Recovery

Enter the command again with a valid argument. You can specify autoflush, autosynch, crmod, debug, localchars, netdata, or options.

`Unknown mode` *`mode`* `('mode ?' for help)`

Cause

The mode you specified is invalid.

Effect

The command is ignored.

Recovery

Enter the command again and specify `c[haracter]` or `l[ine]`.

`WRITEX error (`*`error-number`*`) on term fd` *`file-number`*

Cause

A write error occurred when TELNET tried to write to the specified file.

Effect

The operation fails.

Recovery

See "Recovering From Errors" (page 162).

TFTP Error Messages

This subsection describes the TFTP error messages.

TFTP Client Error Messages

These error messages appear on the client's screen.

`?Ambiguous command`

Cause

The command you specified does not contain enough characters to identify it clearly.

Effect

The command is ignored.

Recovery

Enter the command again, but specify enough characters to identify it clearly.

`?Ambiguous help command`

Cause

The command you specified when requesting help does not contain enough characters to differentiate it from other commands.

Effect

The help command is ignored.

Recovery

Enter the help command again and include enough characters to identify the TFTP command you want described.

`Cannot convert` *`file-name`* `to GUARDIAN internal name`

Cause

In a command that creates a name for a new file from the name of the original file, the original file name cannot be converted to a valid Guardian name. For example, the file name might contain more than eight characters.

Effect

The operation you requested is not performed.

Recovery

Enter the command again and specify a valid Guardian name for the new file.

`discarded` *`number`* `packets`

Cause

TFTP displays the number of packets discarded during a transfer.

Effect

The transfer completes. This is statistical information.

Recovery

No recovery action needed.

`File code mismatch, delete old file`

Cause

You are replacing an existing file by transferring a file of the same name. The file code of the existing file does not match the file code of the new file. If you transfer an ASCII file, the new file is assigned file code 101. A binary file is assigned file code 0.

Effect

The operation is not performed.

Recovery

Delete the existing file and enter the command again.

file-name : TFTP error code *code expansion-of-code*

Cause

An error occurred during the file transfer operation. The possible errors are:

0	Not defined, see error message (if any).
1	File not found.
2	Access violation.
3	Disk full or allocation exceeded.
4	Illegal TFTP operation.
5	Unknown transfer ID.
6	File already exists.
7	No such user.

Effect

The transfer operation fails.

Recovery

Compare the command you entered to the specific error displayed. You might need to correct a file or user name. If encounter have an access violation or a disk is full, you might need to ask the system administrator why a file is unavailable or how to obtain access to a file. Typically, an access violation relates to the security restrictions on the file you want to transfer.

Name does not resolve to supplied parameters.

Cause

You specified a host name that is not known to the system.

Effect

The command is ignored.

Recovery

Verify that you are using the correct host name and enter the command again with the correct name.

?Invalid command

Cause

The command you entered is not a valid TFTP command.

Effect

The command is ignored.

Recovery

Enter a valid TFTP command. You can display a list of valid commands by entering the help command.

?Invalid help command *command*

Cause

You have requested information about a command that is not valid in TFTP.

Effect

The help command is ignored.

Recovery

Enter the help command without any parameters, in order to display information about all TFTP commands.

mode-specified unknown mode
usage: mode [ascii netascii binary image octet]

Cause

The mode you specified is not valid.

Effect

The command is ignored.

Recovery

Enter the command again and specify ascii, netascii, binary, image or octet mode.

No target machine specified.

Cause

You did not specify a remote host when you started TFTP or when you entered a connect command, so you must specify a host in the get or put command.

Effect

The command is ignored.

Recovery

Enter the command again and specify the host for the remote file.

port-num bad port number

Cause

You specified an invalid port number.

Effect

The command is ignored.

Recovery

Specify a port number from 1 through 65535.

tftp: udp/tftp: unknown service

Cause

The $SYSTEM.ZTCPIP.SERVICES file does not include a listing for TFTP.

Effect

TFTP cannot operate.

Recovery

Ask your system manager to enter TFTP in the SERVICES file.

Transfer timed out.

Cause

The remote system failed to respond at the start of or during a transfer operation.

Effect

The operation fails.

Recovery

Try the transfer later. If possible, contact the system administrator of the remote system to determine the status of the remote system.

usage: [get/put] file ...[*host*:]*target*

Cause

The get or put command you entered is not correct. (The host name does not appear if the connection has been established.)

Effect

The transfer operation fails.

Recovery

Make sure that you are using the correct command syntax and specifying valid host and file names.

`usage: [rexmt/timeout] value`

Cause

You have not specified a value with a rexmt or timeout command.

Effect

The command is ignored.

Recovery

Enter the command again and specify a positive number.

`usage: tftp host-name [port]`

Cause

You have entered an invalid RUN command.

Effect

The command interpreter prompt appears.

Recovery

Check the syntax of the TFTP RUN command and enter the command again. Make sure that you specify valid parameters in the correct order.

`value-given: bad value`

Cause

You specified an improper value in the rexmt command.

Effect

The command is ignored.

Recovery

Enter the command again with the correct value. You cannot enter a negative number.

TFTP Server Error Messages

These error messages appear on the operator's screen. They are also logged to $0.

`Bringing down TFTPSRV...`

Cause

The Server terminates because of an exceptional condition. This message is always preceded by an EMS message describing the exceptional condition.

Effect

The server terminates.

Recovery

Take the recommended action for the condition as described in the EMS message.

`Child Process Create Err error-number, Err-detail error-detail`

Cause

Creation of the TFTPCHLD process failed.

The *error-number* field indicates the process-creation error returned by the PROCESS_CREATE_() procedure call. For more information, see the *Guardian Procedure Calls Reference Manual*.

If *error-number* is 1, the *error-detail* field contains a file-system error number. For information about file-system errors, see the *Guardian Procedure Errors and Messages Manual*.

Effect

The TFTPSRV process cannot start the TFTPCHLD process. TFTPSRV sends a negative acknowledgment to the client.

`Child Process is Non-Existent...`

Cause

TFTPSRV cannot communicate with the TFTPCHLD process it created. As a result, a new TFTPCHLD is created.

Effect

None.

Recovery

None. This is an informational message.

`File open error on Child Process. Error` *error-number*

Cause

The TFTCHLD process failed to open. The TFTPCHLD process might have terminated. The *error-number* field indicates the file-system error. For details, see the *Guardian Procedure Errors and Messages Manual*.

Effect

TFTPSRV cannot start the TFTPCHLD process. TFTPSRV sends a negative acknowledgment to the client.

Recovery

None.

```
Getting Startup message for Child Process failed with
Error:-1
```

Cause

The TFTPCHLD process did not receive the start-up message sent by the parent process.

Effect

The TFTPSRV process cannot start the TFTPCHLD process. TFTPSRV sends a negative acknowledgment to the client.

Recovery

None.

`Ignoring excessive Subvolumes`

Cause

The valid volumes and subvolumes names specified during start-up is greater than upper limit.

Effect

The TFTP Client can access files only from the first 10 subvolumes specified.

Recovery

Configure TFTPSRV with 10 subvolumes or fewer.

`Incorrect IP address`

Cause

The IP address specified in the TACL PARAM TFTP^HOST^IP is invalid.

Effect

The TFTPSRV process terminates.

Recovery

Set the TACL PARAM TFTP^HOST^IP to a valid IP address before starting the TFTPSRV process.

`vol.subvol: Invalid Subvolume Name`

Cause

The location specified is not valid. For example, the location name could have more than the permissible 36 character, or the location name might not be specified in the correct format.

Effect

Files cannot be uploaded to or downloaded from the specified location.

Recovery

Determine whether the volume and subvolume was specified in the correct format and whether the subvolume exists on the system.

`tftpd: bind_nw error-number`

Cause

Binding to a the specified port failed. This might be due to another process that is currently occupying the port.

Effect

The TFTPSRV process terminates.

Recovery

Check the value of *error-number*. For an explanation of the *error-number*, see "Errors" under the bind-nw heading in the "Library Routines" section of the *TCP/IP and TCP/IPv6 Programming Manual*.

`tftpd: socket_nw error-number`

Cause

The socket_nw() call failed to create the socket. For an explanation of error-number, see the "Errors" subsection of the description of the socket_nw call in the *TCP/IP Programming Manual*.

Effect

The TFTPSRV process terminates.

Recovery

Make sure that TCP/IP is running. Try restarting the TFTPSRV process.

`tftpd: tftp/udp: unknown service`

Cause

The User Datagram Protocol (UDP) is not started on the system.

Effect

The TFTPSRV process terminates

Recovery

Have your System Administrator check the SERVICES file to determine whether it contains the entry tftp:udp. If it does not, the entry must be added.

`Unable to find Any Valid subvolume`

Cause

All of the *vol.subvol* names specified at startup of the TFTPSRV are invalid.

Effect

The TFTPSRV process will stop after logging an EMS message and the message "Bringing down TFTPSRV...".

Recovery

Start TFTPSRV with at least one valid subvolume.

`Unable to find the default TFTPCHLD object`

Cause

The TFTPCHLD object is neither present in the subvolume from which the TFTPSRV process was started, nor is it present in the default location $SYSTEM.ZTCPIP.

Effect

The TFTPSRV process terminates.

Recovery

When you install TFTPSERV, make sure that the TFTPCHLD object is stored in $SYSTEM.ZTCPIP.

`Unable to find the default subvolume $DATA.PUBLIC`

Cause

The validation of *vol.subvol* specified during start-up, and the server could not find the default volume and subvolume, $DATA.PUBLIC.

Effect

The TFTP Server process stops.

Recovery

Specify at least one valid subvolume during start-up or make sure the default subvolume $DATA.PUBLIC is present on the system.

`Unable to find Volume:` *vol.subvol*`. File sytem error:` *error-number*

Cause

The *vol.subvol* does not exist or the link to *vol.subvol* is down. The *error-number* identifies the file system error associated with the message. For details, see the *Guardian Procedure Errors and Messages Manual*.

Effect

Files cannot be uploaded to or downloaded from *vol.subvol*.

Recovery

Start TFTPSRV with at least one valid subvolume.

`Unable to get the current System Name`

Cause

The TFTP Server process cannot retrieve the system name.

Effect

The TFTP Server process stops.

Recovery

Contact the system administrator.

`Writing Request Packet into Child Failed.`

`Error` *error-number*

Cause

TFTPSRV failed to write the Request Packet to the TFTPCHLD process. The *error-number* field indicates the error. For details, see the *Guardian Procedure Errors and Messages Manual*.

Effect

TFTPSRV stops the TFTPCHLD process. TFTPSRV also sends a negative acknowledgment to the client.

Recovery

None.

`Writing Startup Message to Child Process Failed with Error: error number`

Cause

TFTPSRV failed to write the startup message to the TFTPCHLD process. The *error-number* field indicates the file-system error. For details, see the *Guardian Procedure Errors and Messages Manual*.

Effect

TFTPSRV stops the TFTPCHLD process. TFTPSRV also sends a negative acknowledgment to the client.

Recovery

None.

TN6530 Error Messages

This subsection describes the TN6530 error messages.

`'argument': ambiguous argument ('mode ?' for help)`

Cause

The argument you specified in a mode command does not contain enough characters to identify it clearly.

Effect

The command is ignored.

Recovery

Enter the command again, but specify enough characters to identify it clearly as either character mode or line mode.

`'argument': ambiguous argument ('toggle ?' for help)`

Cause

The argument you specified in a toggle command does not contain enough characters to identify it clearly.

Effect

The command is ignored.

Recovery

Enter the command again, but specify enough characters to identify it clearly as one of the following: autoflush, autosynch, crmod, debug, localchars, netdata, or options.

`'argument': unknown argument ('mode ?' for help)`

Cause

The argument you specified in a mode command is not valid.

Effect

The command is ignored.

Recovery

Enter the command again with a valid argument. You can specify character or line.

`'argument': unknown argument ('toggle ?' for help)`

Cause

The argument you specified in a toggle command is not valid.

Effect

The command is ignored.

Recovery

Enter the command again, but specify a valid argument. You can specify autoflush, autosynch, crmod, debug, localchars, netdata, or options.

`Cannot create status window`

Cause

TN6530 could not create a window.

Effect

TN6530 terminates.

Recovery

See "Recovering From Errors" (page 162).

`Cannot create window`

Cause

TN6530 could not create a status line.

Effect

TN6530 terminates.

Recovery

See "Recovering From Errors" (page 162).

`Conflicting entries found when scanning` *`key-word`*

Cause

The keyboard mapping file (/etc/map6530) contains a conflict. The emulator keyword *`key-word`* contains a sequence that has been defined previously for another emulator keyword in the mapping file.

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Edit the /etc/map6530 file to make sure that no two keywords are defined with the same sequence. See Appendix A, Keyboard Mapping for TN6530 and Appendix C, Installing TN6530 on a Sun Workstation for more information.

`Connection closed by foreign host`

Cause

The remote host closed your TELNET connection.

Effect

Your session with the remote host is terminated.

Recovery

If you did not close the connection, try to make the connection again.

`First character of sequence for` *`key-word`* `is not a control type character`

Cause

In the keyboard mapping file (/etc/map6530), the definition of *`key-word`* does not begin with a control-type character such as a circumflex (^).

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Edit the keyboard mapping file and correct the definition of the *key-word* specified in the message.

key-word: is not allowed to be specified by a user

Cause

The *key-word* is reserved by TN6530.

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Terminate TN6530 and edit the keyboard mapping file by changing *key-word* to a valid key word.

key-word unknown 6530 key identifier

Cause

The specified *key-word* in the keyboard mapping file (/etc/map6530) is not a valid emulator keyword.

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Edit the keyboard mapping file to correct *key-word*. See the tables in Appendix A for a list of valid keywords.

Missing definition part for 6530 key *key-word*

Cause

No definition is provided for *key-word* in the keyboard mapping file (/etc/map6530).

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Edit the keyboard mapping file and supply the missing definition. See Appendix A for more information.

Missing semi-colon for 6530 key *key-word*

Cause

The definition for *key-word* in the keyboard mapping file (/etc/map6530) is not terminated with a semicolon (;).

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Edit the keyboard mapping file and supply the missing semicolon. See Appendix A, Keyboard Mapping for TN6530 for more information.

Required equal sign after 6530 key identifier *key-word* missing

Cause

The definition for *key-word* in the keyboard mapping file (/etc/map6530) is missing an equal sign (=) between the keyword and the scan code.

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Edit the keyboard mapping file and supply the missing equal sign.

`TERM environment variable (that defines the kind of terminal you are using) is not set. To set it, say 'setenv TERM type'`

Cause

TN6530 cannot determine what type of terminal you are using.

Effect

TN6530 continues, but you cannot operate in block mode.

Recovery

Terminate TN6530 if you want to use block mode. Enter the UNIX command setenv TERM *type*. Set the *type* to the type of terminal you are using (for example, setenv TERM SUN).

`Unable to find entry for terminal-type in file filename`

Cause

The terminal type is not defined in the file specified.

Effect

TN6530 continues, but you cannot operate in block mode.

Recovery

Edit the file and correct or add the terminal type.

`Unable to open file filename`

Cause

TN6530 was unable to open the keyboard mapping file.

Effect

TN6530 continues, but its keys are not mapped.

Recovery

Check the status of the keyboard mapping file (/etc/map6530). If the file has been deleted, make a new copy as described in Appendix C, Installing TN6530 on a Sun Workstation.

`Using default key mappings`

Cause

TN6530 was unable to use the specified keyboard mapping.

Effect

TN6530 uses the default mapping scheme.

Recovery

If you want to use some other mapping scheme, check the keyboard mapping file (/etc/map6530) for invalid entries and correct them.

C Installing TN6530 on a Sun Workstation

This appendix provides instructions for installing the TN6530 program on your Sun workstation.

To install TN6530, you transfer files from a NonStop system to your Sun workstation. Follow these steps:

1. Log in as the user named root. You must have system administrator privileges.
2. Run the FTP client on your system and establish a connection with a NonStop host system. For example, to connect to a NonStop host named tandhost, enter the following command:

   ```
   # ftp tandhost
   ```
3. Issue the cd command to change to the subvolume $system.ztcpip:

   ```
   ftp> cd $system.ztcpip
   ```
4. Issue the get command to copy the file map6530 from the NonStop system to your workstation as /etc/map6530:

   ```
   ftp> get map6530 /etc/map6530
   ```
5. Change to binary mode:

   ```
   ftp> bin
   ```
6. Issue the get command to copy the program file from the NonStop system to your workstation as /usr/ucb/tn6530. The name of the program file you copy depends on the type of workstation you are using. Table 23 lists the program files according to workstation type.

   For example, the file name for one Sun/3 workstation is tn6530s3:

   ```
   ftp> get tn6530s3 /usr/ucb/tn6530
   ```

   NOTE: You must name the new file tn6530. If you try to run the program by any other name, you cannot use 653X block mode functions.
7. Disconnect from the NonStop system and exit the FTP client program:

   ```
   ftp> quit
   ```
8. Change the tn6530 file into an executable file:

   ```
   # chmod a+x /usr/ucb/tn6530
   ```

If you want to change the functionality of any keys, you can modify the /etc/map6530 file as described in Appendix A, Keyboard Mapping for TN6530.

Table 23 TN6530 Program Files for Sun Workstations

Program File	Sun Operating System Version	Sun Workstation Model
SUN3OS4 *	SUNOS 4.0.3	Sun 3
SUN4OS4	SUNOS 4.1.1	SUN 4 (SPARCSTATION)
SUNIOS4	SUNOS 4.0.1	Sun 386i

* Does not operate on SUNOS 3.4 Operating System

D TN6530 Control Codes and Escape Sequences

This appendix summarizes the 653X terminal control codes and escape sequences that are supported by TN6530. " TN6530 Control Codes and Escape Sequences" (page 187) lists the control and escape sequences that TN6530 does not support; Table 25 lists the control and escape sequences that are supported.

Table 24 Unsupported Control and Escape Sequences

Control/Escape Sequence	Function
CAN (18H)	Cancel line
ESC - X	Set single-width characters
ESC - Y	Set 80-column format
ESC - Z	Set 132-column format
ESC - 0	Print page
ESC f	Disconnect modem
ESC t	Set double-width characters
ESC x	Set I/O device configuration
ESC y	Read I/O device configuration

Table 25 Supported Control and Escape Sequences

Control/EscapeSequence	Function
BEL (07H)	Sound bell (audible alarm)
BS (08H)	Backspace (cursor left)
HT (09H)	Horizontal tab
LF (0AH)	Line feed (cursor down)
CR (0DH)	Carriage return
DC1 (11H)	Set buffer address*
DC3 (13H)	Set cursor address
ESC (1BH)	Escape sequence header
GS (1DH)	Start field*
ESC - C	Set buffer address extended*
ESC - I	Clear memory to spaces extended*
ESC - D	Set cursor address extended*
ESC - J	Read with address extended*
ESC - K	Read with address all extended*
ESC 1	Set tab
ESC 2	Clear tab
ESC 3	Clear all tabs
ESC 6	Set video attributes

* Sequences used in block mode only

** Sequences used in conversational mode only

Table 25 Supported Control and Escape Sequences *(continued)*

Control/EscapeSequence	Function
ESC 7	Set video prior condition register
ESC :	Select page*
ESC ;	Display page
ESC <	Read buffer*
ESC =	Read with address*
ESC >	Reset modified data tags*
ESC ?	Read terminal configuration
ESC @	Delay one second
ESC A	Cursor up
ESC C	Cursor right
ESC F	Cursor home down
ESC H	Cursor home
ESC I	Clear memory to spaces
ESC J	Erase to end of page/memory
ESC K	Erase to end of line/field
ESC L	nsert line*
ESC M	Delete line*
ESC N	Disable local line editing*
ESC O	Insert character*
ESC P	Delete character*
ESC S	Roll up**
ESC T	Roll down**
ESC U	Page down**
ESC V	Page up**
ESC W	Enter protect submode*
ESC X	Exit protect submode*
ESC [	Start field extended*
ESC]	Read with address all
ESC ^	Read terminal status
ESC _	Read firmware revision level
ESC a	Read cursor address
ESC b	Unlock keyboard
ESC c	Lock keyboard
ESC d	Simulate function key

* Sequences used in block mode only

** Sequences used in conversational mode only

Table 25 Supported Control and Escape Sequences *(continued)*

Control/EscapeSequence	Function
ESC i	Back tab*
ESC o	Write to message field
ESC p	Set max page number*
ESC q	Reinitialize*
ESC r	Define data type table*
ESC u	Define RETURN key function**
ESC v	Set terminal configuration
* Sequences used in block mode only	
** Sequences used in conversational mode only	

E Parameters for the FTP Server Process in the PORTCONF File

You must start the LISTNER process on the system on which the FTP Server runs. The LISTNER process scans the entries in the port configuration (PORTCONF) file and invokes the FTP Server upon receiving a request from the FTP client. For a description of the PORTCONF file, see the *TCP/IP Configuration and Management Manual.*

In the PORTCONF file, you can specify any or all of the following optional parameters after the specification of *port-number*:

```
port-number FTP-server [ / run-options /]

  [ AUTH 0|1 ]

  [ GUARDIAN 0|1 ]

  [ IPSIZE max-size]

  [ LOTS 0|1 ]

  [ TIMER timer-value ]

  [ UEXT x,y,z ]

  [ USERS filename ]
```

port number
: specifies the well-known port number for the FTP Server

FTP-server
: is the name of the FTP server.

[/ *run-options* /]
: are the optional parameters of the TACL RUN command. The only run option supported with FTP is HIGHPIN ON/OFF. For a full description of this run option, see the *TACL Reference Manual*.

NOTE: If you specify this run option, you must do so before you specify the optional parameters described in this appendix. If you interchange the order of these specifications, the FTP Server will not be invoked.

AUTH 0|1
: turns off the user authentication performed by the FTP Server prior to the execution of every command. User authentication is the default behavior. Setting the value of AUTH to 0 disables user authentication. Setting the value to 1 or any numeric value other than 0 results in the default behavior.

  Leaving out the AUTH specification from the PORTCONF file results in the FTP Server's retaining the default behavior.

NOTE: No matter what the AUTH setting is, the FTP Server always authenticates the user when the user logs on.

GUARDIAN 0|1
: selects Guardian as the default FTP Server personality for users who do not log on with aliases (that is, users who log on with user names). Setting the value of GUARDIAN to 1 allows such users to log on with a Guardian personality. Setting GUARDIAN to 0 disables the Guardian personality for such users.

  Leaving out the GUARDIAN specification form the PORTCONF file disables the Guardian personality for users who do not use aliases.

For alias users, the FTP Server will use the default personality according to the INITIAL DIRECTORY attribute specification. For example, if the INITIAL DIRECTORY specification is OSS, the default personality is OSS. If there is no INITIAL DIRECTORY specification, the FTP Server chooses Guardian as the default personality.

NOTE: The INITIAL DIRECTORY attribute is specified as part of the Safeguard configuration. You can view it with the following SAFECOM command:

INFO USER *user-id* DETAIL

FTP default personality selection depends on whether the INITIAL DIRECTORY attribute is set.

If a valid initial directory is set, the user is logged on with an OSS default personality.

If no initial directory is set, the user is logged on with a Guardian personality

.If OSS is not running or if a nonexistent initial directory is set, normal users are logged on with a Guardian personality. The logons of anonymous users, however, are denied in this case.

For more information on the INITIAL DIRECTORY attribute, see the *Safeguard Reference Manual*.

IPSIZE *max-size*

specifies the maximum transfer size used when data is transferred to and from processes. The value for *max-size* is a numeric value, specifying the maximum transfer size in bytes.

LOTS 0|1

specifies that the Last Open TimeStamp be updated for every GET request from the FTP Server. Setting the value of LOTS to to 1 enables the updating of the Last Open TiemStamp. Setting the value to 0 disables such updating.

Leaving out the LOTS specification from the PORTCONF file disables the updating of the Last Open TimeStamp.

TIMER *timer-value*

specifies idle-time-out value for the FTP Server. When you include TIMER in the PORTCONF file, the FTP server times out after the specified time if there is no session activity. The value of *timer-value* is a numeric value defining the time-out interval in seconds.

UEXT *x,y,z*

over-rides the default values for the primary, secondary, and maximum extents used for the transfer of unstructured files.

The value for *x* is a numeric value from 1 through 65535. This value specifies the primary extent size in pages (2048-byte units).

The value for *y* is a numeric value from 1 through 65535. This value specifies the secondary extent size in pages (2048-byte units).

The value of *z* is a numeric value from 1 through 978. This value specifies the maximum number of extents of a local or remote file.

Leaving out the UEXT specification from the PORTCONF file causes the FTP Server to use the default values.

FTP allows users to specify their own primary, secondary, and maximum extent for each data transfer call they make. If the user specifies any of these values within the call, the FTP Server will:

- Use that value
- Ignore the specifications for the other extent values in the PORTCONF file
- Use the defaults for the values for primary extents and maxextents if you specify only secondary extents.

If the user does not specify any extent values in the call, the FTP Server considers the values specified in the PORTCONF file to be the user-defined values.

If the user specifies no extent values in either in the PORTCONF file or within the call, the FTP Server uses the default values.

USERS *filename*

allows you to specify those users who are allowed to log on. The variable specifies the name of a file containing the names of users who are allowed to log on.

If you set USERS with the name of a file that does not exist, the FTP Server allows no one to log on.

If your set USERS but do not specify *filename*, the FTP Server replies with a “Remote server configuration problem” message to any user trying to log on.

Index

Symbols

A

B

C

D

E

F

G

O

P

Q

R

T

U

V

W

X