Telserv Manual

Abstract

This manual describes the HP NonStop™ Telserv subsystem. Part I contains a product overview. Part II contains operational and configuration information for system administrators, operators, support planners. Part III contains information for interactive terminal users. Part IV contains information for programmers.

Product Versions

G06 (G-series), H01 (H-series and J-series)

Supported Release Version Updates (RVUs)

This guide supports the G06.24 RVU and all subsequent G-series RVUs, H06.03 RVUS and all subsequent H-series RVUs, and J06.03 and all subsequent J-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
427174-006	February 2010

Document History

Part Number	Product Version	Published
427174-003	Telserv G06	September 2003
427174-004	Telserv G06	December 2003
424174-005	Telserv G06 (G-series) Telserv H01 (H-series)	July 2005
424174-006	Telserv G06 (G-series) Telserv H01 (H-series and J-series)	February 2010

Telserv Manual

Glossary Index Figures Tables

4. Subsystem Control Facility (SCF) for Telserv

5. Subsystem Control Facility (SCF) Commands for Telserv

5. Subsystem Control Facility (SCF) Commands for Telserv (continued)

C. SCF Command Syntax (continued)

Tables

What's New in This Manual

Manual Information

Abstract

This manual describes the HP NonStop™ Telserv subsystem. Part I contains a product overview. Part II contains operational and configuration information for system administrators, operators, support planners. Part III contains information for interactive terminal users. Part IV contains information for programmers.

Product Versions

G06 (G-series), H01 (H-series and J-series)

Supported Release Version Updates (RVUs)

This guide supports the G06.24 RVU and all subsequent G-series RVUs, H06.03 RVUS and all subsequent H-series RVUs, and J06.03 and all subsequent J-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
427174-006	February 2010

Document History

Part Number	Product Version	Published
427174-003	Telserv G06	September 2003
427174-004	Telserv G06	December 2003
424174-005	Telserv G06 (G-series) Telserv H01 (H-series)	July 2005
424174-006	Telserv G06 (G-series) Telserv H01 (H-series and J-series)	February 2010

New and Changed Information

This manual has been updated to reflect support of a new PARAM that is used to bind Telserv 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 Table 3-1, Telserv Parameter Names and Values in the TACL PARAM Command, on page 3-3.

In addition, support of the CIP subsystem has been added throughout the manual, where ever NonStop TCP/IP processes are discussed. The only technical difference from a Telserv perspective pertains to network partitions. This is explained in Special Consideration for Users of Logical Network Partitions and PROVIDERS on page 3-13.

About This Manual

This section includes the following information:

- Who Should Read This Guide
- How This Guide Is Organized
- Related Documentation
- Notation Conventions
- Notation Conventions

The *Telserv Manual* provides information about the HP TELNET Server (Telserv) subsystem.

Who Should Read This Guide

This guide is addressed to three types of users:

- System administrators, operators, and support planners responsible for the proper operation of the Telserv subsystem
- Users at interactive terminals who access Telserv
- Programmers who write service applications and client software that interface to the Telserv subsystem

How This Guide Is Organized

Part I, “Overview,” introduces the Telserv subsystem. It contains:

- Section 1, The TELNET Server (Telserv) discusses the functionality and principal components of the Telserv subsystem.

Part II, “Configuration and Management,” provides information for system administrators, operators, and support planners responsible for the configuration and proper operation of the Telserv subsystem. It contains:

- Section 2, Configuration Quick Start shows the minimum amount of tasks you must perform to make the Telserv subsystem operational.
- Section 3, Starting the Telserv Process shows the TACL RUN command used to start the Telserv process and lists all available RUN options. It also lists the parameter names and parameter values used with the TACL PARAM command to determine the environment in which the Telserv process operates.
- Section 4, Subsystem Control Facility (SCF) for Telserv gives an overview of SCF and discusses the SCF object types for the Telserv subsystem.

- Section 5, Subsystem Control Facility (SCF) Commands for Telserv lists all SCF commands, objects, and object attributes for the Telserv subsystem.
- Section 6, Using PTrace contains the subsystem-specific details for using PTrace to format Telserv subsystem trace files created through the use of the SCF TRACE command.

Part III, "Accessing and Using Telserv from a Terminal," provides information for interactive users at terminals or terminal emulators who access Telserv to perform tasks.

- Section 7, Accessing and Using Telserv Services provides information for those who access Telserv from terminals and terminal emulators to perform specific tasks. This section includes a discussion of modes of operation.
- Section 8, Using TELNET to Connect to Telserv shows the commands used to connect the TELNET client to the Telserv process.
- Section 9, TELNET Client Commands and TELNET Commands lists TELNET client commands and TELNET commands.

Part IV, "Programming Information," provides information for programers who write service-application software or client software that interfaces to the Telserv process.

- Section 10, Modes of Operation provides information about modes of operation for programmers.

Three appendixes provide reference information.

- Appendix A, Telserv Error Messages Sent to Terminals lists error messages that the user of a terminal or terminal emulators can receive.
- Appendix B, SCF Error Messages for Telserv lists error messages that the SCF user can receive.
- Appendix C, SCF Command Syntax shows the command syntax for the SCF commands used to configure and manage the Telserv subsystem.

Related Documentation

To use Telserv, you should be familiar with the following HP manuals:

- *Guardian User's Guide*
- *Guardian Procedure Errors and Messages Manual*
- *Guardian Procedure Calls Reference Manual*
- *Distributed Systems Management (DSM) Manual*
- For G-series: *Introduction to Networking for HP NonStop S-Series Servers*
- For H-series: *Introduction to Networking for HP NonStop NS-Series Servers*
- For J-series: *NonStop Networking Overview*

- *PTrace Reference Manual*
- *TACL Reference Manual*
- *Operator Messages Manual*
- *OSM Users' Guide*

For information about TCP/IP and TELNET, you can consult the following manuals:

- *TCP/IP Applications and Utilities User Guide*

  For G-series: *TCP/IP Configuration and Management Manual*

  TCP/IP (Parallel Library) Configuration and Management Manual

  TCP/IPv6 Configuration and Management Manual

  For H-series and J-series: *TCP/IP Configuration and Management Manual*

  TCP/IPv6 Configuration and Management Manual

  Cluster I/O Protocols (CIP) Configuration and Management Manual

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

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
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

`file-name`

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. *`Italic computer type`* letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

```
pathname
```

[ ] 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.
```

lowercase italic letters. Lowercase italic letters indicate 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
```

Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

Part I. Overview

1 The TELNET Server (Telserv)

For HP NonStop™ systems, the HP Telserv subsystem is the implementation of the server portion of the TELNET protocol. The TELNET protocol is a general, bi-directional, 8-bit, byte-oriented protocol in the TCP/IP protocol suite. It specifies how terminal devices and terminal-oriented processes must interface.

Telserv runs on a NonStop system as an operating-system process and uses the socket library routines of the TCP/IP software for TCP access. One Telserv process supports up to 256 virtual terminals or “windows” through which a user’s data can be sent and received after undergoing transformation to TELNET semantics.

As an HP product, Telserv consists of two components:

- The TELNET server process (called the Telserv process)
- The TELNET client process that runs on the NonStop system.

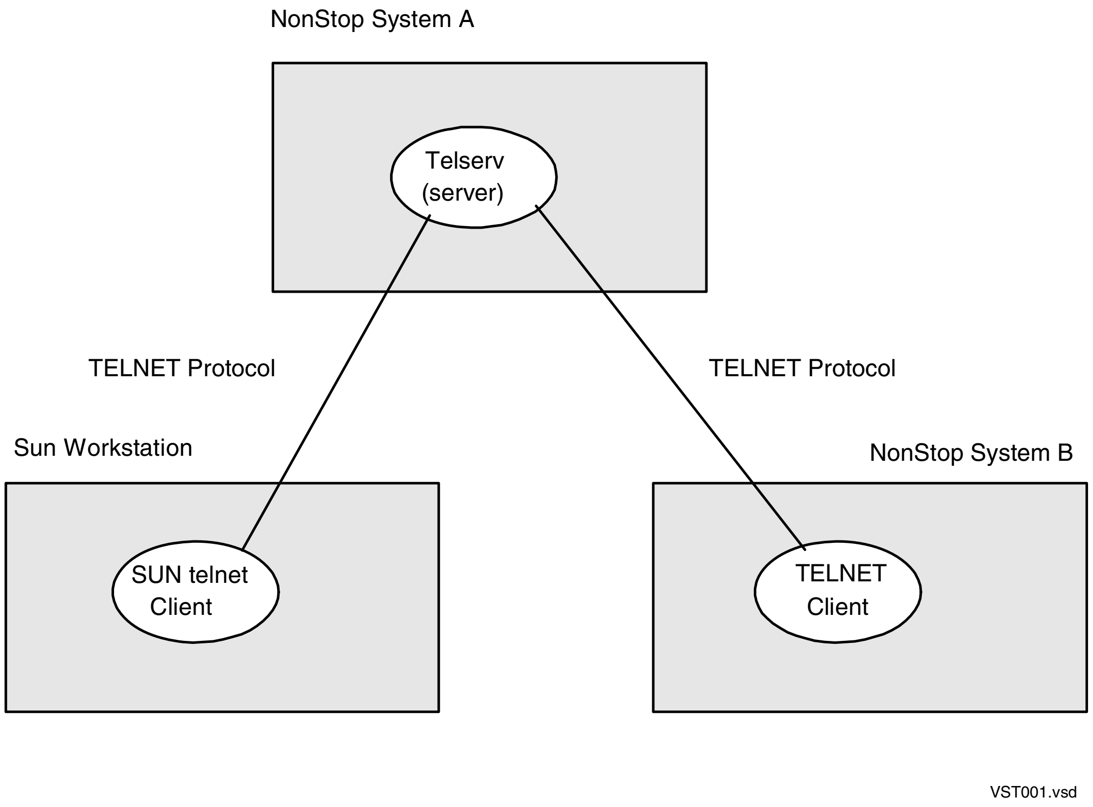
Other vendors offer terminal emulators containing client software that allows various types of workstations to access Telserv. The Telserv subsystem can provide services to:

- HP TELNET clients hosted on NonStop systems
- Clients included in terminal emulators that use the TELNET protocol to communicate with host NonStop systems
- Generic clients that support the TELNET protocol

The client features included in the HP TELNET client differ from the features other vendors provide. While other vendors’ products might support the 6530 terminal type, the HP TELNET client runs in NVT mode only.

Figure 1-1 shows the Telserv process on a NonStop S-series server A communicating with two client applications. NonStop server B shows an HP TELNET client communicating with Telserv. This figure also shows a non-HP client application communicating with Telserv. This non-HP client could use Telserv to access host processes or files.

Figure 1-1. TELNET Server (HP Telserv)



Telserv is configured to run over one of the NonStop implementations of TCP/IP: NonStop TCP/IP, Parallel Library TCP/IP, NonStop TCP/IPv6 or Cluster I/O Protocols (CIP).

Note. H-series and J-series systems do not support Parallel Library TCP/IP. G-series systems do not support CIP.

Additionally, PCT and TTE users on G-series systems have the choice of connecting to a Telserv process that is configured over the IPX/SPX subsystem. IPX/SPX is not available on H-series and J-series systems.

Telserv can be configured to run in the NonStop Guardian environment, or in the NonStop Open System Services (OSS) environment. OSS is the HP NonStop POSIX offering.

Interactive Interface

The Telserv subsystem is managed by entering interactive (SCF) commands that affect the operation of subsystem objects. For information about the Telserv SCF interface, see the following sections in Part II, “Configuration and Management:”

- Section 4, Subsystem Control Facility (SCF) for Telserv
- Section 5, Subsystem Control Facility (SCF) Commands for Telserv

Programmatic Interface

The DSM programmatic interface allows user-written application programs to perform the same management functions as those available through SCF. Like SCF, DSM applications depend on SCP to communicate with the different subsystems. DSM applications provide the additional capability of automating many of the management tasks that can be useful in a complex computing environment.

Event messages are sent directly from the subsystem to the EMS event collector ($0). You can select (filter) the types of event messages to be displayed, printed, or stored.

Part II. Configuration and Management

2 Configuration Quick Start

Starting the Telserv subsystem requires only a single operation. Simply issue a TACL RUN command to create the Telserv process.

Before issuing the RUN command, make sure that the underlying protocol stack is up and running. You can also (but need not) use the TACL PARAM command to specify certain parameters that determine operational aspects of the Telserv process (see Telserv Parameter Names and Values for the TACL PARAM Command on page 3-1). If you do not specify these parameters, Telserv operates using their default values.

The RUN command also provides RUN options that determine the operation of the Telserv process. Using the RUN Command and Specifying Run Options on page 3-9 describes these options. If you do not specify them, Telserv operates using their default values.

The following diagram shows the syntax of a RUN command sufficient to start the Telserv process and make it operational. This command starts a single Telserv process with no backup. The diagram assumes that the Telserv process is being started within the volume and subvolume in which the Telserv subsystem was installed.

```
TACL> RUN TELSERV/NAME $process-name, NOWAIT/ [ port-number ]
```

$process-name

specifies the name of the Telserv process you will run.

[*port-number*]

specifies the port number. You can run multiple Telserv processes simultaneously by specifying unique port numbers for each process.

When the Telserv process is up and running, you can use the Subsystem Control Facility to configure and manage the Telserv environment.

To configure the Telserv environment, you must be aware of the hierarchy of objects within the Telserv subsystem.

The PROCESS object refers to the Telserv process. The Telserv PROCESS object has two subordinate objects:

- The SERVICE object is subordinate to the PROCESS object.
- The WINDOW object is subordinate to the SERVICE object.

One PROCESS supports as many as 256 services and 256 windows.

The SERVICE object allows users to configure and inquire about aspects of their TELNET session. By adding a SERVICE object, you can control aspects of the client sessions that connect through that particular SERVICE. For example, you can:

- Only provide access to specified users or user groups

- Launch specified programs
- Provide recovery from network and system failures (called a resilient session)

The WINDOW object is the virtual terminal through which a user's data can be sent and received after undergoing transformation to HP NonStop TELNET semantics. The WINDOW object provides the interface between a client application and the Telserv process.

For a full description of these objects and the commands used to configure them, see Subsystem Control Facility (SCF) Commands for Telserv on page 5-1.

3 Starting the Telserv Process

Port 23 is the well-known port that is reserved for TELNET communication. When you start an HP Telserv process, if you don't specify a port number in the RUN command, the process uses port 23 by default. To run multiple Telserv processes per TCP/IP process, you must configure each additional Telserv process to communicate over another port. Note that the TELNET clients must be configured so that they use the port number of the particular Telserv process with which you want them to communicate.

The Telserv process provides a file-system interface to the application layer of TCP/IP and, IPX/SPX.

Note. The interface to IPX/SPX is available on G-series systems only.

Telserv Parameter Names and Values for the TACL PARAM Command

Before starting a Telserv process, you can use TACL PARAM commands to specify parameter names and values that control operational aspects of the process.

For example you can choose the underlying transport protocol that the Telserv process is to use. You can specify either:

- Any one of the available TCP/IP processes, or
- IPX/SPX processes

The following command specifies the underlying transport protocol:

```
PARAM ZTNT^TRANSPORT^PROCESS^NAME [\system.]$name
```

`where $name` specifies the name of the transport process.

Table 3-1 lists all parameter names and values supported for the HP Telserv process.

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 1 of 7)

Name	Description
ZTNT^ATP^COMPATIBLE	Enables or disables certain features similar to those in the ATP6100 product. Values are YES or NO. The default value is NO. The syntax is: **PARAM ZTNT^ATP^COMPATIBLE (YES | NO)** Setting this PARAM to YES enables the following features: ● In Block mode, the WRITE data portion of a WRITEREAD procedure call has an EOT appended to it. (An EOT is normally appended only to the data portions of a WRITE or READ call or to the READ data portion of a WRITEREAD call.) ● A CONTROL-12 operation from any opener of a window results in the termination of the session, unless another opener has break-only access. (Normally, a session is not terminated upon receipt of a CONTROl-12 request unless the opener is the last remaining one.) If this PARAM is set to NO or allowed to default, these features are not enabled. Setting this PARAM to YES is not recommended under normal circumstances because: ● The EOT appended in the WRITREAD call results in Telserv not strictly conforming to the 6530 protocol. ● The termination of a session because of a CONTROL-12 request can cause security problems.

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 2 of 7)

Name	Description
ZTNT^CONN^ON^MAXTTY	Specifies whether new connections are accepted when Telserv is handling the maximum number of sessions. The default value is YES. The syntax is: **PARAM ZTNT^CONN^ON^MAXTTY (YES | NO)** Setting the PARAM value to YES or allowing it to default causes Telserv to continue accepting new connections and to send an error message when MAXTERMINALS is reached. Setting the PARAM value to NO causes Telserv to stop accepting new connections when the MAXTERMINALS limit is reached. New connections are then accepted only after the number of sessions falls below the MAXTERMINALS limit. Telserv sends no error message indicating why it refused a connection request.
ZTNT^CUSTOM^BANNER	Specifies whether or not Telserv sends a customized welcome banner instead of the default welcome banner. Values are YES or NO. The default value is NO. The syntax is: PARAM ZTNT^CUSTOM^BANNER (YES | NO) Setting this PARAM value to YES causes Telserv to read-in the contents of an EDIT file (code: 101) named TSBANNER at start-up time. The file must be located in the same subvolume in which the Telserv executable file resides. The file can contain a maximum of 50 lines. If Telserv cannot find the file or read it, Telserv continues the initialization process, but does not display a banner, either customized or default. If this PARAM value is set to NO or allowed to default, Telserv sends the default welcome banner.

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 3 of 7)

Name	Description
ZTNT^ECHO^SPL^CHAR	Allows the Telserv process to suppress the echoing of Special Characters (Line Kill, End Of File, and Erase Character) after the DONT ECHO option is negotiated. Setting PARAM ZTNT^ECHO^SPL^CHAR to OFF enables this behavior. The syntax is: `PARAM ZTNT^ECHO^SPL^CHAR (OFF|ON)` The DEFAULT value of this PARAM is ON. When set to ON, Special Characters are echoed even if DONT ECHO is negotiated. This PARAM is applicable only when the Telserv process is in DONT ECHO mode. When Telserv process is in ECHO mode, this PARAM does not cause a behavior change.

ZTNT^HOST^IP	Binds Telserv to an IP address.
	The syntax is:
	`PARAM ZTNT^HOST^IP` ***IP-address***
	The IP address assigned must be a valid numerical network address, that is, an IPv4 address in dotted-decimal format or an IPv6 address in hexadecimal format.
	Examples are:
	`PARAM ZTNT^HOST^IP 192.168.10.10`
	`PARAM ZTNT^HOST^IP 3ffe:1200:214:1:a00:8eff:fe04:6ef2`

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 4 of 7)

Name	Description
ZTNT^KEEPALIVE^OFF	Specifies whether to disable the KEEPALIVE socket option.Values are YES or NO. The default value is NO. The syntax is: **PARAM ZTNT^KEEPALIVE^OFF (YES I NO)** Setting the PARAM value to YES causes Telserv to continue the KEEPALIVE function. Setting the PARAM value to NO or allowing it to default causes Telserv not to set the KEEPALIVE socket option for all sockets. Use caution when disabling the KEEPALIVE option. Because the setting is on a process level, it will affect other clients that rely on “keep-alive” for recovery and other purposes. Note that Telserv will not be notified of all terminal disconnections. Therefore, use the idle timer or the banner inactivity timer or both to close sessions after a fixed period of time. Use SCF to configure both the idle timer and the banner inactivity timer. Note that in SCF, the idle timer is displayed as Timeout Value and the banner inactivity timer is displayed as Banner Timeout value.

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 5 of 7)

Name	Description
ZTNT^LOG^CONNECTS	Enables or disables the generation of EMS messages when a client connects to or disconnects from Telserv. Values are YES or NO. The default value is NO. The syntax is: **PARAM ZTNT^LOG^CONNECTS (YES | NO)** Setting the PARAM value to YES causes Telserv to generate an EMS connect event message when a client requests a connection and a corresponding disconnect message when the client disconnects. Telserv sends the event messages to the standard EMS collector ($0). Each event message contains the following ● Window name ● Client IP address ● Service name ● Local-port identifier ● The timestamp for the connection and disconnection Because the number of such event messages can overload the standard EMS collector, you can use another PARAM along with this one to divert the messages to an alternate connector. (For a description of this PARAM, see ZTNT^LOG^ CONNECTS^COLLECTOR.) When the PARAM value is set to NO or is allowed to default, Telserv does not generate connect or disconnect event messages. Note that Telserv continues to generate all other event messages, regardless of the setting for ZTNT^LOG^CONNECTS.
ZTNT^LOG^CONNECTS^ COLLECTOR	Diverts connection and disconnection EMS event messages to the alternate EMS collector that you specify. Use this PARAM in conjunction with ZTNT^LOG^ CONNECTS The syntax is: **PARAM ZTNT^LOG^CONNECTS^COLLECTOR *alt-EMS-coll*** Note that EMS messages other than connection and disconnection messages continue to be sent to the standard EMS collector ($0).

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 6 of 7)

Name	Description
ZTNT^LOG^SERVICENAME	Displays the service name chosen, in the CONNECT and DISCONNECT EMS event messages, in the case of login failure. Use this PARAM in conjunction with ZTNT^LOG^CONNECTS. The syntax is: PARAM ZTNT^LOG^SERVICENAME (YES|NO). The DEFAULT value is NO. When the PARAM value is set to NO or is allowed to default, Telserv displays the default service name (ZTELNET) for CONNECT and DISCONNECT event messages in the case of login failure
ZTNT^SENDBRK^ON^DISCONN	Specifies whether or not a break message is sent to the break owner when a session terminates. Values are YES or NO. The default value is NO. A break message is sent to the break owner only if all of the following conditions are true: ● This PARAM is set to YES. ● There is no outstanding request. ● There is only one opener and that opener is a Guardian application. ● The window is not a resilient window. Sending a break message should cause the break owner to issue an I/O and, after receiving an error 140 message, to perform an appropriate recovery. In the case of TACL, this takes approximately ten seconds.

Table 3-1. Telserv Parameter Names and Values in the TACL PARAM Command (page 7 of 7)

Name	Description
ZTNT^SERVICECHOICE^ECHO	Enables or disables the echoing of user input before Telserv accepts the service chosen by the user. The Values are YES or NO. The default value is YES. The syntax is: **PARAM ZTNT^SERVICECHOICE^ECHO (YES I NO)** Setting this PARAM value to YES or allowing it to default causes Telserv to echo user input. Setting this PARAM value to NO disables the echoing of the input the user generates while choosing a service. Disable echoing when both of the following situations exist: ● Telserv is configured with services that have the DISPLAY attribute turned off, and ● For security reasons, the user selection should not be displayed on the terminal. Note that normal echoing will be restored after Telserv accepts the user's service choice.
ZTNT^TACL^PROGRAM	Specifies the file name of a TACL program other than the default. When TACL is chosen, this alternate TACL program is used.
ZTNT^TRANSPORT^PROCESS^NAME	Specifies the name of the TCP/IP or IPX/SPX process to be used as the underlying transport protocol. The syntax is **PARAM ZTNT^TRANSPORT^PROCESS^NAME [\*sysname*].$procname** Use of this feature depends on the availability of the Socket library (T9550) on the local system. Telserv will not operate unless T9550 is present on the local system.

Using the RUN Command and Specifying Run Options

You create the HP Telserv process by issuing a TACL RUN command.

Note. The LOGIN program must reside on the same subvolume as Telserv. Because the LOGIN program contains privileged code, it must be licensed. The Telserv process must be started by super.super to support OSS or unauthenticated services other than TACL.

Following is the RUN command syntax for creating a Telserv process. Note that, if line breaks are necessary when you enter the command string, it is good practice to insert an ampersand (&) at the end of each broken line (as shown in the example following the syntax information).

```
RUN $volume.subvolume.TELSERV /NAME $process-name, NOWAIT,
  [ HIGHPIN OFF, ]
  [ CPU cpu-number,] [ PRI priority ]/  [-8 ] [ port-number ]
  [ -BACKUPCPU cpu-number     ]
  [ -BREAKDATA | -NOBREAKDATA ]
  [ -CTRLECHO | -NOCTRLECHO   ]
  [ -FLOWCTRL | -NOFLOWCTRL   ]
  [ -KANJI                    ]
  [ -LINEMODE                 ]
  [ -NOBANNER                 ]
  [ -NOEOFMSG                 ]
  [ -NOMENU                   ]
  [ -NOPROMPT                 ]
  [ -NOTACL                   ]
  [ -XTABS | -NOXTABS        ]
```

`$`*`volume.subvolume`*`.TELSERV`

specifies the location where Telserv is installed. Unless you are starting Telserv from within the volume and subvolume in you installed Telserv, you must specify `$`*`volume`* and *`subvolume`*.

`$`*`process-name`*

specifies the name of the Telserv process you will run.

`[ HIGHPIN OFF, ]`

specifies that the process is to run at a low-PIN.

By default, the process runs at a high-PIN.

`[ CPU` *`cpu-number`* `]`

specifies the CPU number of the primary process.

`[ PRI` *`priority`* `]`

specifies the process-run priority. PRI should never be set lower than 170.

`[ -`*`bit-number`* `]`

where:

`-8`

indicates that data is transferred between the server and client in 8-bit format and in binary code. The default value is 8.

`-7`

indicates that data is transferred between the server and client in 7-bit format and in ASCII code. Therefore, this format cannot be used when you are transferring binary data.

If you do not specify this option, the 8-bit format is used.

If the client negotiates terminal type with Telserv and the terminal type name is TN6530, the data is transferred in 7-bit format regardless of the default data transmission.

If the client terminal type name is TN6530-8, the data is transferred in 8-bit format regardless of the default transmission.

`[ `*`port-number`*` ]`

specifies the port number. You can run multiple Telserv processes simultaneously by specifying unique port numbers for each process.

When communicating with TCP, the default is port 23.

When communicating with SPX, the default is port `0x7000`.

The value can be specified in either decimal or hexadecimal notation, regardless of the transport process.

`[ -BACKUPCPU `*`cpu-number`*` ]`

specifies the CPU number of the backup process. When you specify a backup CPU, the Telserv process runs as a NonStop process pair.

If you do not specify this option, no backup CPU is assigned.

`[ -BREAKDATA | -NOBREAKDATA ]`

determines whether or not CTRL-C is interpreted as a BREAK character or as a regular data character. SETMODE 263 provides the programmatic interface.

For information about SETMODE procedures, see the *Guardian Procedure Calls Reference Manual*.

`[ -CTRLECHO | -NOCTRLECHO ]`

determines whether or not non-printing characters are converted to printable characters on the terminal display. SETMODE 262 provides the programmatic interface.

For information about SETMODE procedures, see the *Guardian Procedure Calls Reference Manual*.

`[ -FLOWCTRL | -NOFLOWCTRL ]`

determines whether or not CTRL-S and CTRL-Q are interpreted as flow control characters or as regular data characters. Note that you use CTRL-S and CTRL-Q to pause and restart rapidly scrolling text on the terminal. SETMODE 264 provides the programmatic interface.

For information about SETMODE procedures, see the *Guardian Procedure Calls Reference Manual*.

`[ -KANJI ]`

makes the backspace (BS) key operational when you use TELSERV in binary mode (for example, with clients that work in the Kanji environment). Each keystroke erases one character from the terminal display. To erase a Kanji character, which is double-byte, use 2 BS keystrokes.

If you do not specify this option, the BS key is not operational when TELSERV is in binary mode.

`[ -LINEMODE ]`

establishes TELNET sessions in line mode. All sessions are established in line mode unless you are using TN6530/UB6530 clients. Such clients are not configured for X6530 line mode. The advantages of using line mode are:

- Reduced network traffic
- Faster response time.

If you do not specify this option, sessions are not established in line mode.

`[ -NOBANNER ]`

suppresses display of the Welcome Banner.

If you do not specify this option, the Welcome Banner is displayed.

`[ -NOEOFMSG ]`

prevents TELSERV from generating the EMS message “Socket read error... EOF detected”. (Telserv EMS Message #8).

In a large system that has many LAN controllers, a processor failure could result in the generation of a large number of these messages, which, in turn, could overload the EMS collector process ($0). This option prevents such an overload.

If you do not specify this option, EOF messages are generated.

`[ -NOMENU ]`

suppresses display of the Service Menu.

If you do not specify this option, the Service Menu is displayed.

`[ -NOPROMPT ]`

suppresses display of the selection prompt, which is used to choose an item from the Service Menu.

If you do not specify this option, a selection prompt is displayed.

`[ -NOTACL ]`

specifies that TACL is not displayed as one of the services in the selection menu.

If you do not specify this option, TACL appears as a service on the selection menu.

`[ -XTABS | -NOXTABS ]`

determines whether or not TAB characters are converted to spaces. SETMODE 261 provides the programmatic interface.

For information about SETMODE procedures, see the *Guardian Procedure Calls Reference Manual*.

The following is an example of the command for creating a Telserv process:

```
RUN $TESTER.SYSTEM.TELSERV /NAME $ZTNT, NOWAIT, &
   HIGHPIN OFF, CPU 0,PRI 170/ -7 5555 -BACKUPCPU 1 &
   -NOBANNER -NOMENU -NOPROMPT -NOTACL -XTABS &
   -CTRLECHO -BREAKDATA -FLOWCTRL -NOEOFMSG -KANJI
```

Special Consideration for Users of Logical Network Partitions and PROVIDERS

If you are using the Logical Network Partitioning (LNP) feature of NonStop TCP/IPv6, you must run a Telserv process for each indexed logical network partition through which you want Telserv to service connections. Use the DEFINE and PARAM commands to perform this task.

For example, the following command sequence starts a Telserv process on a default logical network partition and another on an indexed logical network partition:

Default LNP:

```
>PARAM TCPIP^PROCESS^NAME $ZSAM1

>PARAM ZTNT^TRANSPORT^PROCESS^NAME $ZSAM1

>TELSERV/TERM $ZHOME, OUT $ZHOME, NAME $ZTN1,&
CPU 0, NOWAIT, PRI 170/ -BACKUPCPU -1
```

Indexed LNP:

```
>PARAM TCPIP^PROCESS^NAME $ZB019

>PARAM ZTNT^TRANSPORT^PROCESS^NAME $ZB019

TELSERV/TERM $ZHOME, OUT $ZHOME, NAME $ZT019, &
CPU 0, NOWAIT, PRI 170/ -BACKUPCPU 1
```

In CIP, network partitions are created through the SCF PROVIDER object. Similarly to NonStop TCP/IPv6, you must run a Telserv process for each PROVIDER through which you want Telserv to service connections. Use the same DEFINE and PARAM command for PROVIDERs as you use for LNPs. The TCP/IP process name shown in the example for LNPs, $ZSAM1 and $ZB019 would be the CIP TPNAMEs associated with the SCF PROVIDER object instead of the NonStop TCP/IPv6 socket access method processes. Both the TCP/IP processes configured for LNP in NonStop TCP/IPv6 and the TPNAME of the PROVIDER object configured in CIP are TCP/IP transport service providers that have been configured to be restricted to particular IP addresses.

For complete information on logical network partitions, see the *TCP/IPv6 Configuration and Management Manual*. For information about configuring PROVIDERs, see the *Cluster I/O Protocols (CIP) Configuration and Management Manual.*

Fault Tolerance

When you specify the -BACKUPCU option, the Telserv process runs as a NonStop process pair. Failure of the primary process terminates all connections to Telserv, and all current users must reconnect. The backup process takes over and becomes the primary process. As the new primary process, it attempts to create a new backup for itself.

If the takeover was due to process failure, this new primary process creates its backup in the CPU in which the failed process was running.

If the takeover was due to CPU failure, the new primary process does not select the next available CPU to launch its backup, but rather creates its backup when the failed CPU is reloaded.

Whenever a takeover occurs, Telserv makes the configuration data (specified through SCF or through a management program) available to the backup process taking control. Thus, you need not reconfigure the Telserv process manually. Configuration data, when input to the Telserv process, is saved to an ENSCRIBE, key-structured file. When taking over, the backup process reads the file in order to replicate the configuration of the original primary. This file, or configuration database, resides in the same subvolume as the Telserv process program file.

Each process pair has its own configuration database with a unique file name. To ensure uniqueness, the three-letter string ZTZ is prefixed to the process-pair name. For example, a process pair with the name $ZTNT acquires a configuration database named ZTZZTNT.

Telserv does not delete the configuration database when you stop the process. You must therefore periodically remove unused databases. If you start a Telserv process and a configuration database with a corresponding name already exists, Telserv overwrites the original database.

Altering and Stopping a Telserv Process

Once a Telserv process has been started with the RUN command, you can display information on the process, alter it, or stop it with the Telserv Subsystem Control Facility (SCF) commands. You can alter, for example, the maximum number of windows allowed, control the display of the service menu, and change the time after which inactive windows (client sessions) are logged off. Section 4, Subsystem Control Facility (SCF) for Telserv introduces the Telserv SCF interface.

4

Subsystem Control Facility (SCF) for Telserv

This section provides background information essential to managing the HP Telserv subsystem interactively through the use of the Subsystem Control Facility (SCF). Topics covered here include:

- General information on SCF operation
- Prerequisites
- Compatibility information.
- Definition and discussion of SCF objects for Telserv

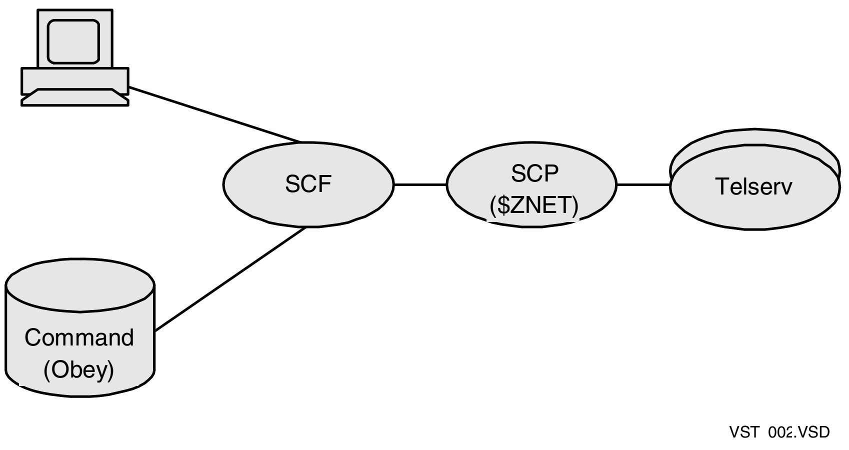
SCF Operation

SCF is used to configure, control, and collect information about NonStop data communications subsystems.

When you generate a new system, the system configuration is established for devices such as disk drives, tape drives, and data communications lines. However, depending on the subsystem being configured, device configuration that is performed during the generation process typically does not supply all the information needed to configure certain communications entities such as subdevices (terminals and printers). For data communications subsystems, you establish these additional configuration details using SCF.

SCF provides an operator interface to an intermediate process called the Subsystem Control Point (SCP). SCP in turn provides an interface to the various I/O processes performing data communications operations and services. The relationship between these elements is illustrated in Figure 4-1. The default SCP process, known to the system as $ZNET, provides access for application programs, whether local or remote, to monitor and control a communications subsystem.

Figure 4-1. SCF Overview



Subsystem Control Point (SCP)

The Subsystem Control Point (SCP) is a network-management process for receiving and redistributing the messages that SCF sends to certain data communications subsystems like Telserv.

SCP lies between SCF and Telserv. SCP provides security (by restricting access to sensitive commands), version control, and tracing support for subsystems, as well as providing support for application processes.

SCF automatically opens and closes SCP. In most cases, the default SCP is the only one you will ever need. If you should need to establish an SCF session through a specific SCP other than the default SCP, you can start additional SCPs by using either the SCF RUN command or the TACL RUN command. For more information about SCP, see the *SCF Reference Manual for SCP*.

SCP can be used in two ways: interactively and programmatically. The interactive interface is provided so you can choose either to let a person perform an action or to automate the action.

The programmatic interface is based on the Subsystem Programmatic Interface (SPI), which provides procedures that build and retrieve information from command, response, and event-message buffers. SPI is described in the *Distributed Systems Management (DSM) Programming Manual*.

SCF Commands

Several commands are available for displaying and changing SCF session parameters. Table 4-1 shows these commands and their effects.

Table 4-1. Changing SCF Session Parameters

Command	Effect
ASSUME	defines a default object to be used when the object is omitted from an SCF command.
ENV	displays the current settings of the SCF command parameters that establish the program environment.
HELP	displays a list of the available SCF commands. You can also request additional specific information, such as command syntax, for each command. For most subsystems, you can also request subsystem-specific information for their commands. The HELP key is also available, as it is in TACL, to display command syntax.
OBEY	causes commands to read from a a specified command file in interactive mode.
OUT	directs the display output to a specified file (the output file), while continuing to send display output to the location set at SCF startup time.
SYSTEM	designates the default system name for all file-name and object-name expansions.
VOLUME	designates the default volume and subvolume for expansion of all file-names.
RUN	allows you to run another program during an SCF session.

Other SCF commands operate on the objects (lines, subdevices, processes, and so forth) belonging to each subsystem. In the Telserv subsystem, for example, the ADD and DELETE commands add objects to and delete objects from the list of objects controlled by SCF. Once an object is subject to control by SCF, you can use the START, STOP, and ABORT commands to change the state of the object, or you can use the ALTER command to change the values of selected attributes of the object. You can also use the INFO command to display the current attribute values for objects and the STATUS command to display the current dynamic status of objects.

Note. For a complete description of the commands described in this section, refer to the *Subsystem Control Facility (SCF) Reference Manual*. General SCF commands are not described in this manual.

Sensitive and Nonsensitive Commands

Because some commands can have detrimental effects if improperly used, special qualification is required to use them. These commands are called sensitive commands. A sensitive command is one that can be issued only by a user with super-group access, the owner of the subsystem, or a member of the same group as the owner of the subsystem. When used in conjunction with the security features of the NonStop

Kernel operating system, SCF provides effective access control for sensitive commands. Commands that request information or status but do not affect operation are called nonsensitive commands. Nonsensitive commands are available to all users.

Objects

SCF controls a wide variety of data communications subsystems whose individual components are called objects. Each object has an object type and an object name. The object type describes the type of the object, such as LINE, SU (subdevice), or PROCESS. The object name uniquely identifies an object within the system.

Object names usually follow a consistent set of naming conventions. Objects defined during system generation are usually referred to by their device names, which consist of a dollar sign ($) followed by a letter and from zero through six alphanumeric characters. A LINE object, for example, might have the name $LIN32. A PROCESS object sometimes has the same name as the line it controls.

Subdevices are often not defined during system generation and must be added to the subsystem by using the SCF ADD command. The object name for an SU object usually has two parts: the name of the line to which the SU has been added and the unique name of the subdevice itself. The subdevice part of the name begins with a pound sign (#) and includes up to seven alphanumeric characters. For example, an SU object might have the name $LIN32.#PC, where $LIN32 is the name of the line.

There are many other object types and object names used in SCF. The objects relevant to Telserv are described later in this section.

Subordinate Objects

Some subsystems are structured hierarchically, with a group of objects of one type logically subordinate to objects of another type. For example, a number of subdevices could be configured on a single line. In this case, an object specification (*object-spec*) of the form *line-name** (as in $LINE*) can be used to refer to all subdevices on that line. Some SCF commands include a SUB parameter that refers to all subordinate objects.

The SUB parameter has the following form:

```
SUB [ subtype ]
```

This parameter selects the subordinate objects affected by the command. *subtype* can be a subordinate object type or one of the keywords ONLY or ALL. If a subordinate object type is named, only objects of that type are affected. ONLY specifies that only the subordinate objects are affected. ALL specifies that the named object and the subordinate objects are affected. If *subtype* is omitted, ALL is assumed.

States

Objects can have operational states, such as STOPPED or STARTED. The exact sequence of states an object goes through varies from object to object and from subsystem to subsystem. SCF commands for the Telserv subsystem recognize only two states: STOPPED and STARTED. Applicable states for Telserv are discussed below.

The operational state of an object at a given instant is important. For example, some commands have no effect on objects when those objects are in the STOPPED state, but can affect the object when it is in the STARTED state.

The following states are recognized by the Telserv subsystem:

State	Description
STOPPED	The object is not ready for normal operations. STOPPED is equivalent to down, not ready, or killed.
STARTED	The object is initialized and ready for normal data traffic.

How SCF Works

For commands that relate only to the SCF session (such as VOLUME), SCF takes the appropriate action without communicating with SCP. For commands that relate to a subsystem or its objects, SCF translates the command into a formatted message for SCP, which then communicates with the appropriate subsystem to perform the specified task.

SCF accepts commands from a terminal, a disk file, or an application process. It sends display output to a terminal, a file, a process, or a printer. When SCF is started interactively, the input source and output destination are specified in the command-interpreter RUN command used to start SCF. If SCF is started by a process-creation procedure, its input source and output destination are taken from the startup message. Subsequent SCF commands can change the input source and output destination.

Input Sources

SCF accepts command input from a terminal or a disk file. The initial input source is determined by the form of the RUN command used to initiate SCF. At any time during an SCF session, the input source can be temporarily changed to execute a series of commands from a command (Obey) file.

Modes of Operation

SCF can be run in two modes: interactive or noninteractive. Because interactive and noninteractive input are treated differently by SCF, the following distinctions are important:

- The mode is interactive when both input and output pass through the same terminal, or when the same process is used for both input and output. For instance, using the OUT command or using the `/ OUT` *`file-spec`* `/` parameter within a command causes SCF to run in noninteractive mode. This rule affects the use of abbreviations when you enter keywords: abbreviations (formed by truncating keywords, provided the result is unambiguous) are permitted in interactive mode only. For example, in interactive mode you can abbreviate the VERSION command to VE, but in a command file (noninteractive mode) you must spell out VERSION.

- Interactive input from a terminal is characterized by the entry of a command followed by a carriage return. When a process is used for both input and output, SCF waits for the process to send a request and treats the process in the same manner as a terminal.
- Noninteractive input usually appears in the form of a command (Obey) file. Command (Obey) files are usually EDIT files that contain a series of commands.

For more information about interactive and noninteractive modes, see the *Subsystem Control Facility (SCF) Reference Manual*.

Setting the Initial Input Source

To specify an initial input source for an SCF session, use the IN option of the RUN command. You can specify a terminal (identified by its logical-device name) or a disk file. For example, the following TACL RUN command initiates SCF and directs it to read commands from a disk file named $DATA.SCF.STARTUP (if not specified, the system name and volume name are the default names currently in use by the TACL command interpreter through which the RUN command was entered):

```
19> SCF / IN $DATA.SCF.STARTUP /
```

If you run SCF from the TACL command interpreter without specifying an input file, SCF assumes that the input is coming from your terminal.

Output Destinations

SCF can direct output to a disk file, an application process, a terminal, or a printer. The initial output destination is determined by the form of the RUN command used to initiate SCF. The output destination can be changed dynamically during an SCF session.

Setting the Initial Output Destination

To specify an initial output destination for an SCF session, use the OUT option of the RUN command. You can specify a terminal (identified by its logical-device name), a disk file, an application process, or a printer. For example, the following RUN command initiates SCF and directs it to send its output to a disk file named $DATA.SCF.DISPLAY (if not specified, the system name and volume name are the default names currently in use by the TACL command interpreter through which the RUN command was entered):

```
20> SCF / OUT $DATA.SCF.DISPLAY /
```

If you run SCF without specifying an output file, SCF assumes that the output is being sent to your terminal.

Running SCF

You can start SCF interactively by using the TACL command interpreter RUN command or by using a command file (or TACL macro or routine). The RUN command used to start SCF is described in this section and in the *TACL Reference Manual*.

All the SCF commands are described in Appendix C, SCF Command Syntax.

SCF usually resides in a file named $SYSTEM.SYSTEM.SCF.

General Command Format

An SCF command always begins with a keyword identifying the command (such as, START, VOLUME, or TRACE).

If the command pertains only to the current SCF session, the keyword is followed immediately by whatever additional parameters are required to specify the action to be taken by SCF. For example, the SYSTEM command can be entered as SYSTEM \newsys, where *newsys* is the name of the system that is to become the default system.

If the command pertains to an object, it is followed immediately by the object type and the object name. For example, the following command aborts the LINE object called $LIN32:

```
ABORT LINE $LIN32
```

This command aborts a subdevice on that line:

```
ABORT SU $LIN32.#PC
```

If additional information is required, the object name is followed by a comma and the parameters required to further specify the action to be taken. For example, the following command changes the value of the BCUG attribute for a line:

```
ALTER LINE $LIN32, BCUG 200
```

SCF commands for Telserv permit the use of the command parameter SUB. You can use SUB to specify which subordinate objects the command affects. For example, you may want to get information for all the Telserv objects subordinate to a Telserv process. One way to do this is with the following command:

```
INFO PROCESS $ZNT, SUB ALL
```

Specific information about the format of each SCF command appears in Appendix C, SCF Command Syntax.

SCF Online Help

SCF provides online support when you use the HELP command described in the *Subsystem Control Facility (SCF) Reference Manual*. If the HELP command alone is entered, SCF responds with a menu that guides you through the available help options. Select from the options displayed in the menu for the information you need.

Using this menu-mode help facility, you can obtain general information about basic SCF components, such as attribute specifications, character strings, commands, file names, integers, and so forth.

In addition to getting general SCF information, you can obtain information specific to your particular subsystem.

Note. The online help facility is interactive; it cannot be invoked from a command (Obey) file.

SCF Help Utility

The SCF interface to the Telserv subsystem includes online help. Table 4-2 lists the commands you can issue and the type of information returned by the SCF help utility.

Table 4-2. SCF Online Help Commands

This Help Command...	Returns This Telserv Information...
HELP TELSERV	Returns generic information about the Telserv subsystem, including lists of supported commands and objects.
HELP TELSERV *object*	Describes information about the Telserv subsystem as it pertains to that particular object.
HELP TELSERV *command*	Describes the specified command as it is used in the Telserv subsystem.
HELP TELSERV *command object*	Describes how the specified command is applied to the specified object in the Telserv subsystem.
HELP TELSERV *error*	Displays the error message corresponding to the specified error number. The display includes an explanation of the error message.

Prerequisites

The Telserv subsystem requires the NonStop TCP/IP subsystem, the Parallel Library TCP/IP subsystem, the NonStop TCP/IPv6 subsystem, or the NonStop IPX/SPX subsystem. The Telserv subsystem and the TCP/IP subsystem each run one process. After creating the process, you define virtual terminals by adding windows.

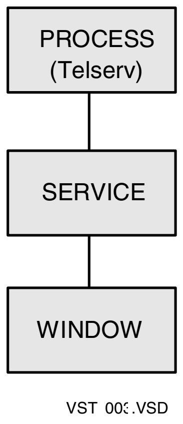
Note. H-series and J-series systems do not support Parallel Library TCP/IP or NonStop IPX/SPX.

Starting the Telserv Process on page 3-1 describes how to start the Telserv process.

SCF Object Types for Telserv

The Telserv subsystem supports three object types: PROCESS, SERVICE, and WINDOW. The SERVICE object is subordinate to PROCESS and WINDOW is subordinate to SERVICE as shown in Figure 4-2.

Figure 4-2. Telnet Object Hierarchy



There can be one or more Telserv processes per subsystem, a maximum of 256 windows per process, and a maximum of 256 service objects per process.

Each of the SCF commands that Telserv is prepared to accept can be used to manipulate objects of one of these three object types.

When you enter an SCF command, you enter the command itself and usually you add the object type and the object name specifier. For many commands, the Telserv subsystem accepts object name templates (text strings containing wild-card characters), giving you a convenient alternative to specifying the exact name of the object or of indicating that multiple or all objects of a given type are to be affected. For information on object name templates, see Object-Name Templates on page 4-13.

Depending on the action of the command, you may also be required to provide an attribute specifier (attribute name and value) to indicate which of the object's attributes are to be changed and what the new value is to be. Details on the required syntax for each command you can use with the Telserv subsystem are provided in Appendix C, SCF Command Syntax.

PROCESS Object

A PROCESS is a subsystem that implements the Telserv protocol.

Naming Conventions for the Object

Starting the Telserv process involves specifying a name that you will later use in your management commands. The name you assign must conform to the following conventions

When you issue the TACL RUN command, the object name you assign to the process object must be unique for all processes on the same NonStop system.

The name must start with a dollar sign ($), and can be followed by up to five alphanumeric characters. The first character following the dollar sign must be a letter. For example:

```
$A1234
```

If the process is to be accessed from remote NonStop systems over an Expand network, the name must start with the dollar sign ($) and can be followed by four alphanumeric characters. The first character following the dollar sign must be a letter. For example:

```
$A123
```

SERVICE Object

A SERVICE allows users to configure and inquire about aspects of their Telserv service.

Naming Conventions for the Object

Adding SERVICE objects involves specifying names that you will later use in your management commands. The names you assign must conform to the following conventions

When you assign an object name to a service object, you must assign a unique name to each service using the same Telserv process.

The name can be up to eight alphanumeric characters in length. The first character must be a letter. The name “EXIT” and names starting with “Z” are reserved for internal use. For example:

```
S1234567
```

A service name is fully qualified to the Telserv subsystem by preceding it with the process name the service is to be associated with. For example:

```
$ZNT01.SERVICE1
```

WINDOW Object

The WINDOW object is the virtual terminal through which a user’s data can be sent and received after undergoing the appropriate transformation to Telserv semantics.

The WINDOW object provides the interface between a client application and the Telserv process.

Naming Conventions for the Object

Adding WINDOW objects involves specifying names that you will later use in your management commands. The names you assign must conform to the following conventions

When you assign an object name to a window object, you must assign a unique name to each window using the same Telserv process. Windows associated with different Telserv processes running on the same NonStop system can have the same window name.

The name must start with a pound sign (#), and can be followed by up to seven alphanumeric characters. The first character following the pound sign must be a letter. For example:

```
#A123456
```

It is recommended that you use the letters WIN followed by a terminal number to identify a window. For example:

```
#WIN05
```

The Telserv process uses the letters PTY when it dynamically creates windows. To avoid conflicts, you should not use the letters PTY followed by a terminal number to identify a window.

A window name is fully qualified to the Telserv subsystem by preceding it with the process name the window is to be associated with. For example:

```
$ZNT01.#WIN05
```

Window Types

Telserv provides three types of window: static, dynamic, and SU.

Static Window

Static windows are those configured through SCF as subordinate to a SERVICE object. Telserv allocates one of the available static windows from the service pool to a client who selects a valid service. Thus, a given client might not get the same window for each session it establishes.

To configure a static window, use a non-null string for the SERVICENAME attribute of the WINDOW object. For example:

```
SCF> ADD WINDOW #win, SERVICENAME “myserv”
```

Multiple static windows can have the same service name.

Clients can access static windows only by specifying the service name (for example, myserv). When a user enters a static window name at the Enter Choice> prompt, Telserv displays the following error message:

```
Sorry, window cannot be selected
Sorry, invalid choice
```

Telserv then displays the selection menu again.

Dynamic Window

Dynamic windows are automatically created and assigned for a session when a service is chosen. Telserv randomly generates the window name, and thus, no WINDOW object need be configured. The SERVICE object, however, must have the subtype specified as DYNAMIC.

Whenever TCP/IP reports a fatal error, dynamic windows return an file-system error 66 message to the application. Telserv then deletes all resources associated with the session. Thus, applications using dynamic windows should not try to re-open the window when they receive an error 66 message. Applications should terminate upon receiving such a message.

Additional Windows

When Telserv reaches the limit of permissible dynamic windows it can create, it can still provide as many as five additional windows. These windows allow users to select static services.

If a user attempts to select dynamic services on one of these windows, Telserv displays an error message and closes the terminal. If a user selects a service whose subtype is STATIC, Telserv processes the request.

A three-minute banner idle time is set for each of the five windows. If a user does not select a static service within three minutes, Telserv closes the window after displaying a warning message.

Note that Telserv does not provide the five additional windows if the command PARAM ZTNT^CONN^ON^MAXTTY NO was issued. In such cases, Telserv closes the window exceeding the MAXTTY limit after it displays the error message “No more sessions are available.” Telserv no longer listens on its port and listens again only when the number of active sessions falls below the MAXTTY limit.

SU Windows

SU windows support applications that must keep the context of the previous session when the client reconnects. To configure an SU window, use a null string for the SERVICENAME attribute of the WINDOW object. For example:

```
ADD WINDOW #win, SERVICENAME ""
```

Only one SU window exists for each Telserv process.

Clients can access SU windows only by specifying the window name (for example, #win). An SU window emulates the behavior of ATP6100-over-dial up-lines in support of 6530 terminal emulators and 6530 terminals attached to the Ungerman-Bass terminal server. SU windows differ from static windows in that Telserv does not reject a connection from a remote client when an SU window is not opened by the application. Such connections are rejected in the case of a static window.

Whenever TCP/IP reports a fatal error, Telserv returns an file-system error 140 message (FEMODEMERR) for all outstanding requests. However, it keeps its internal control block open, preserving the content of the previous session. Consequently, the application can retry the I/O operation without reopening the window. When applications must maintain security, they should authenticate the remote client before retrying the I/O operation or repainting the screen.

Summary of Window Types

Table 4-3 summarizes the differences among static, dynamic, and SU windows.

Table 4-3. Window Types

	Static Window	**Dynamic Window**	**SU Window**
SERVICENAME specification in SCF ADD command	Quoted string	N/A	Null String
Client connects using	Service Name	TACL, etc.	Window Name
File-system error returned to application	140	66	140
Expected application recovery action	Re-open the window and start new session	Terminate	Authenticate client before restarting I/O

Object-Name Templates

Object-name templates allow you to specify multiple objects by entering a pattern composed of a single wild-card symbol or of text and one or more wild-card symbols. With Telserv, you can use the following wild-card symbols in object-name templates:

* Use the asterisk to represent a character string of undefined length. You can use an asterisk to represent the following:

- A trailing string; for example, $ZTNT.#WI* selects all objects subordinate to $ZTNT that have names starting with #WI.
- An undefined number of characters; for example, $ZTNT*5 selects all names that start with $ZTNT and end with 5.

? Use the question mark to represent an unknown character in a specific position; for example, $ZTNT.#PT?Y selects all object names that are subordinate to $ZTNT and begin with #PT, end with Y, and contain exactly four characters after the pound sign.

You can use wild-card characters in any combination. For example:

- $ZTNT.*1 represents all objects that are subordinate to $ZTNT and have some number of characters that precede a 1 and contain no characters following the 1.
- $ZTNT.*1?? represents all objects that are subordinate to $ZTNT and have some number of characters that precede a 1 and contain exactly two characters following the 1.

If you have set a default process name by using the ASSUME command, you can omit the process name and use the asterisk (*) to specify all windows under the assumed process. For example, the following two commands set the default process to $ZTNT and display information about all windows under $ZTNT:

```
-> ASSUME PROCESS $ZTNT
-> INFO WINDOW *
```

The SCF commands you can use with the Telserv subsystem, and the syntax of the SCF commands as they apply to the Telserv subsystem are described in Appendix C, SCF Command Syntax

5 Subsystem Control Facility (SCF) Commands for Telserv

Table 5-1 lists the SCF commands you can use with the HP Telserv subsystem. The table shows the object types to which each command applies.

Table 5-1. SCF Commands for Telserv

Command	PROCESS	SERVICE	WINDOW	*null*
ABORT	X		X	
ADD		X	X	
ALTER	X	X	X	
DELETE		X	X	
INFO	X	X	X	
LISTOPENS	X		X	
NAMES	X	X	X	X
START			X	
STATS	X	X	X	
STATUS	X		X	
STOP	X		X	
TRACE	X		X	
VERSION	X			X

You can enter the following command to obtain detailed syntax information on the Telserv object types and the commands that apply to them:

```
HELP TELSERV [ command ] [ object-type ]
```

The SEL and SUM options, which apply to several of the SCF commands when used with other communications subsystems, cannot be used with the Telserv subsystem.

The term `assumed` is used in the examples in this section to refer to the object specifier that has been specified to the ASSUME command.

The rest of this section describes the syntax of the SCF commands as they apply to the Telserv subsystem.

If you specify `/ OUT file-spec /` with a command, the output generated by the command is directed to the specified file.

Commands can be divided into two categories: sensitive and nonsensitive. Sensitive commands can change the state of the specified object; nonsensitive commands cannot. In addition, sensitive commands can be issued only by users in the SUPER group or issued by any valid user on the system.

ABORT Command

The ABORT PROCESS command forces the Telserv subsystem to shut down, without considerations for data integrity. All outstanding requests are returned with the File System error 201. The sockets are closed and the specified process terminates abnormally (ABEND).

The ABORT WINDOW command terminates the operation of a WINDOW as quickly as possible without regard for the interruption of data flow—only enough processing is done to ensure the integrity of the subsystem. The WINDOW is left in the STOPPED summary state. You can use an object-name template with the ABORT WINDOW command. ABORT WINDOW is a sensitive command.

```
ABORT [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The ABORT command has the following *object-spec*:

```
object-type          object-name

PROCESS               process-name
WINDOW                window-name
```

ABORT PROCESS Command

The syntax for the ABORT PROCESS command is:

```
ABORT [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ SAVEABEND [ ON | OFF ] ]
```

ABORT WINDOW Command

The syntax for the ABORT WINDOW command is:

```
ABORT [ / OUT file-spec / ] WINDOW window-name
```

Considerations for the ABORT Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

PROCESS Object

- Use the STOP PROCESS command for more controlled termination of a process.
- Use the SAVEABEND option to specify whether a saveabend file should be created. Specify ON to create a saveabend file. Specify OFF to suppress the creation of a saveabend file. The default value is OFF.

WINDOW Object

- ABORT WINDOW is a sensitive command.
- Use the STOP WINDOW command for more controlled termination of a window.
- Use the START command to initiate the operation of a window.
- Use the STATUS command to determine the current summary state of the window.

Examples of the ABORT Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS object

The following commands define the assumed process and abort all windows under the assumed process.

```
ASSUME PROCESS $ZTNT
ABORT WINDOW *
```

WINDOW object

The following command aborts a window named $ZTNT.#WIN03:

```
ABORT WINDOW $ZTNT.#WIN03
```

ADD Command

The ADD command adds a SERVICE or a WINDOW to the Telserv subsystem. When the ADD WINDOW command completes, the window is placed in the STOPPED summary state. The Telserv process must already exist before you can ADD a service or window. See Starting the Telserv Process on page 3-1, for details about using the RUN command to create a Telserv process. See Objects on page 4-4 for instructions on naming a service or window. ADD is a sensitive command.

```
ADD [ / OUT file-spec / ] [ object-spec ]

   { , attribute-spec [ , attribute-spec ] ... }
```

The ADD command has the following *object-spec*:

```
object-type          object-name

SERVICE              service-name
WINDOW               window-name
```

ADD SERVICE Command

The *attribute-specs* for the SERVICE object are:

```
[ ACCESS { ALL | SYSTEM | OWNER | NONE } ]
[ ASSIGNED { ON | OFF }                  ]
[ AUTODELETE { ON | OFF }                ]
[ CPU { 0 through 15 }                   ]
[ DEFAULT { ON | OFF }                   ]
[ DISPLAY { ON | OFF }                   ]
[ LIB file-name                          ]
[ OWNER "24-char"                        ]
[ PARAM "128-char"                       ]
[ PRI { 0 through 190 }                  ]
[ PROGRAM file-name                      ]
[ RESILIENT { ON | OFF }                 ]
[ SUBTYPE { DYNAMIC | STATIC }           ]
[ SWAP file-name                         ]
{ TYPE { BLOCK | CONVERSATION | PRINT }  }
```

`ACCESS`

secures a SERVICE access against unauthorized access. For TYPE BLOCK and TYPE CONVERSATION options, the default is SYSTEM. For the TYPE PRINT option, the default is ALL. Specify NONE to prohibit any users from access to the service. This option might be useful during system testing or configuration. If the ACCESS attribute is set to OWNER, the OWNER attribute should be specified; otherwise, the command is rejected with Telserv error 4.

`ASSIGNED`

allows a client to connect directly to a preconfigured STATIC window, based on the IP address of the client. The ASSIGNED attribute is intended to be used in conjunction with the WINDOW attribute, CALLER. The ASSIGNED attribute is available only in an environment where the host Telserv process uses TCP/IP as its underlying transport protocol. Also, it is not functional when used with a RESILIENT service. In these cases, the command is rejected with Telserv error 4. The default value is OFF.

`AUTODELETE`

automatically deletes a SERVICE when all WINDOWS associated with that service are deleted. The default is OFF.

`CPU`

identifies the CPU to use to NEWPROCESS the *`file-name`* identified in PROGRAM. If CPU is not specified, Telserv communicates with CMON to determine which CPU to use. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4.

`DEFAULT`

allows the user to specify a default SERVICE to be automatically accessed when a connection is established, bypassing the welcome menu and service selection options. The default value is OFF.

Each Telserv process can have only one DEFAULT service. If an attempt is made to specify more than one DEFAULT service the command is rejected with Telserv error 19.

`DISPLAY`

controls the display of service names in the Service Menu and controls the amount of SERVICE information provided on #ZSPI. This feature is provided for SERVICES that are security-sensitive. The default is ON.

`LIB`

identifies a local library file name to be used to launch the *`file-name`* identified in PROGRAM. If not specified, the program is launched without a library. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4. Invalid or inaccessible file names are rejected with Telserv error 11.

`OWNER`

identifies the user name (group.user) that owns the service. When this attribute is specified, a client requesting a service is authenticated and access to the service is enforced by the Login Server. Invalid user names are rejected with Telserv error 1.

`PARAM`

contains a run-time parameter string up to 128 bytes to be passed in the STARTUP message of the program being launched. You can enter any string except N/A. If PROGRAM is not specified, this attribute is invalid and is rejected with Telserv error 4.

`PRI`

specifies the program’s priority, otherwise, Telserv uses its own priority minus one. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4.

`PROGRAM`

contains the local file name of the initial program to be launched by the Login Server when a session is established. Invalid or inaccessible file names are rejected with Telserv error 11. Invalid file codes (file code does not equal 100) are rejected with Telserv error 12.

`RESILIENT`

allows a Telserv client to recover from session failures. It is not supported on a DYNAMIC service and is rejected with Telserv error 4. The default is OFF. There can be one STATIC window per resilient service.

`SUBTYPE`

specifies the service subtype. STATIC is the default value.

`DYNAMIC`

defines a service that is not associated with a configured window, for example, a window that was added. A dynamic service automatically creates a window when a session is established.

`STATIC`

allows one or more windows to be added to a given service. STATIC is the default value.

`SWAP`

identifies a local volume name to use as the swap file for the PROGRAM *file-name*. If not specified, the Login Server uses a swap file determined by the system. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4. Invalid or inaccessible SWAP *file-names* are rejected with Telserv error 11.

`TYPE`

specifies the class of service. `TYPE` is a required service attribute. CONVERSATION is the default value.

`BLOCK`

configures a T6530 block mode service and initiates the session in block mode. If this attribute is selected, SETMODE 28 resets the terminal to block mode.

`CONVERSATION`

configures a conversational terminal service and initiates the session in conversation mode.

`PRINT`

configures a printer (write-only device) service.

ADD WINDOW Command

The *`attribute-specs`* for the WINDOW object are:

```
[ CALLER "IP4-address"                              ]
[ CALLER6 "IP6-address"                             ]
[ ENDOFFILE 0 through 255                           ]
[ ENDOFLINE 0 through 255                           ]
[ ERASE 0 through 255                               ]
[ INTERRUPT 0 through 255                           ]
[ LINEKILL 0 through 255                            ]
[ SERVICETYPE { BLOCK | CONVERSATION | PRINT } ]
{ SERVICENAME "service-name" }
```

`CALLER`

allows a client to connect directly to a preconfigured STATIC window, based on the IPv4 address of the client. The CALLER attribute should be used in conjunction with the SERVICE attribute, ASSIGNED. The CALLER attribute is required if the corresponding SERVICE object, to which the window belongs, has its ASSIGNED attribute ON. Otherwise, it is optional. Refer to Considerations for the ADD Command, WINDOW Object, for more information.

`CALLER6`

is a quoted IPv6 address of the Telnet client. This address is automatically assigned to this configured window upon connection. The IP address is specified in eight 16-bit hexadecimal numbers. For example:

```
"5f1c:ce00:df3f:b210:0030:0700:306a:e3e3"
```

Leading blanks in quoted string are invalid. If you specify leading blanks, Telserv will return an error 1.

`ENDOFFILE`

is a 1-byte ASCII value used to signify an end-of-input condition to a reader. The default value is CTRL-Y (%H19 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`ENDOFLINE`

is a 1-byte ASCII value used to signify that an input line is complete. The default value is a CARRIAGE-RETURN (%H0C in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`ERASE`

is a 1-byte ASCII value used to erase a single character of input. The default value is a BACKSPACE (%H08 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`INTERRUPT`

is a 1-byte ASCII value used to send a break message to the process that owns the break on the window. The default value is a CTRL-C (%H03 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`LINEKILL`

is a 1-byte ASCII value used to delete the line being entered. The default value is CTRL-X (%H18 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`SERVICENAME`

is the name of the service the window will be associated with. This name may be any quoted string of printable characters except for reserved names. Reserved names are “EXIT” and names starting with Z. The HP NonStop naming convention for this value is an 8-character string, the first character of which is alphabetic. Programs associated with a given service can be then run on any STARTED window associated with that service. The service name appears in the Telserv banner menu sent to remote users of Telserv clients.

`SERVICETYPE`

is an enumerated value for the type of service with which this window will be associated. CONVERSATION is the default value.

`BLOCK`

indicates that the Telserv initial mode will be block mode; it can subsequently be changed to conversation mode by using the SET MODE command.

`CONVERSATION`

indicates that the Telserv initial mode will be conversation mode; it can subsequently be changed to block mode by using the SET MODE command.

`PRINT`

indicates that the SERVER may initiate a connection, thus allowing a printer (write-only device) to be configured.

Considerations for the ADD Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

All Objects

- ADD is a sensitive command.

SERVICE Object

- A PROGRAM attribute is required for DYNAMIC services.
- The SERVICENAME name must be unique to the PROCESS object. The name For DEFAULT ON:
  - The ASSIGNED attribute should be OFF.
  - The RESILIENT attribute should be OFF.
  - The SUBTYPE attribute can be either STATIC or DYNAMIC.
    - For SUBTYPE STATIC (with DEFAULT ON), Telserv starts presenting the service selection menu after all existing windows under the service are exhausted. Subsequent connections are mapped to dynamic windows.
  - The TYPE attribute should not be PRINT.
- You can add a maximum of 256 services to each Telserv process.

SERVICE Object

The following example adds a service.

```
ADD SERVICE $ztnt.payroll, program $user.pwy.paytcp,&
param "abcdefg", owner "mygroup.myname",access group
```

WINDOW Object

The following example adds a block mode window:

```
ADD WINDOW $ZTNT.#BLK,SERVICENAME "BLK", SERVICETYPE BLOCK
START WINDOW #BLK
```

The following example adds a conversation mode window:

```
ADD WINDOW $ZTNT.#CONV, SERVICENAME "CONV", &
   SERVICETYPE CONVERSATION
START WINDOW #CONV
```

The following example adds a window for a printer connection:

```
ADD WINDOW $ZTNT.#PR1,SERVICENAME "PR1", SERVICETYPE PRINT
START WINDOW #PR1
```

ALTER Command

The ALTER command changes attribute values associated with the specified PROCESS, SERVICE, or WINDOW. The ALTER SERVICE command can only be issued when all subordinate windows are in the STOPPED summary state. The ALTER WINDOW command can only be issued when the window is in the STOPPED state. ALTER is a sensitive command.

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

   { , attribute-spec [ , attribute-spec ] ... }
```

The ALTER command has the following `object-spec`:

```
object-type           object-name

PROCESS               process-name
SERVICE               service-name
WINDOW                window-name
```

ALTER PROCESS Command

When a PROCESS object is created (see Prerequisites on page 4-8), it has five configurable attributes that you can change using the ALTER command. These attributes are `MAXTERMINALS`, `MENU`, `TIMEOUTVALUE`, `BANNERTIMEOUTVALUE,` and `CPULIST`.

The *`attribute-specs`* for the PROCESS object using the ALTER PROCESS command are:

```
[ MAXTERMINALS { 1 through 256 }         ]
[ MENU { ON | OFF }                      ]
[ TIMEOUTVALUE { 3 through 32767 }       ]
[ BANNERTIMEOUTVALUE { 3 through 32767 } ]
[ CPULIST {0,1,2,...,15}]
[ DROPCR { ON | OFF } ]
```

`MAXTERMINALS`

specifies the maximum number of virtual terminals (windows) that can be present in the system. The default value is 256.

`MENU`

controls the display of the service menu. This attribute preempts the setting of the `DISPLAY` attribute for the SERVICE object. The default value is ON.

`TIMEOUTVALUE`

is the time (in minutes) after which inactive terminals are disconnected. If a terminal had been logged on, it is logged off and disconnected. The value 32767 indicates infinite time. If inactive terminals are not to be disconnected, 32767 should be specified. The default value is 32767.

`BANNERTIMEOUTVALUE`

is the time (in minutes) after which a terminal can remain inactive following presentation of the banner in the dynamic window. When the value specified in the `BANNERTIMEOUTVALUE` attribute is reached the terminal is disconnected. The value 32767 indicates infinite time. The default value is 32767.

The current value of `BANNERTIMEOUTVALUE` can be examined using the INFO PROCESS, DETAIL command.

`CPULIST`

specifies the processors used for launching the services according to the round-robin algorithm implemented in the Telserv process. Applications are launched on the processors specified only if the SERVICE object does not specify a CPU attribute and if $CMON is not available. By default, the CPULIST attribute contains all 16 processor numbers. When the processors configured are not available, Telserv attempts to launch the application on the processors that are available.

`DROPCR`

controls the response for Guardian [WRITE]READ[X] calls to termination characters like carriage returns (CR) embedded in text entered by the user on the terminal. The default setting is ON, indicating that null responses are not generated for embedded termination characters.

ALTER SERVICE Command

The *attribute-specs* for the SERVICE object are:

```
[ ACCESS { ALL | SYSTEM | OWNER | NONE } ]
[ ASSIGNED { ON | OFF }                  ]
[ AUTODELETE { ON | OFF }                ]
[ CPU { 0 through 15 }                   ]
[ DEFAULT { ON | OFF }                   ]
[ DISPLAY { ON | OFF }                   ]
[ LIB file-name                          ]
[ OWNER "24-char"                        ]
[ PARAM "128-char"                       ]
[ PRI { 0 through 190 }                  ]
[ PROGRAM file-name                      ]
[ RESILIENT { ON | OFF }                 ]
[ SUBTYPE { DYNAMIC | STATIC }           ]
[ SWAP file-name                         ]
[ TYPE { BLOCK | CONVERSATION | PRINT }  ]
```

`ACCESS`

secures a SERVICE access against unauthorized access. It is invalid if OWNER is not specified and is rejected with Telserv error 4. NONE prohibits any users from access to the service. This option might be useful during system testing or configuration.

`ASSIGNED`

allows a client to connect directly to a preconfigured STATIC window, based on the IP address of the client. The ASSIGNED attribute is intended to be used in conjunction with the WINDOW attribute, CALLER. The ASSIGNED attribute is available only in an environment where the host Telserv process uses TCP/IP as its underlying transport protocol. Also, it is not functional when used with a RESILIENT service. In these cases, the command is rejected with Telserv error 4. The default value is OFF.

`AUTODELETE`

automatically deletes a SERVICE when all WINDOWS associated with that service are deleted.

`CPU`

identifies the CPU to use to NEWPROCESS the *file-name* identified in PROGRAM. If CPU is not specified, Telserv communicates with CMON to determine which CPU to use. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4.

`DEFAULT`

allows the user to specify a default SERVICE to be automatically accessed when a connection is established, bypassing the welcome menu and service selection options. The default value is OFF.

Each Telserv process can have only one DEFAULT service. If an attempt is made to specify more than one DEFAULT service the command is rejected with Telserv error 19.

For DEFAULT ON, consider the following:

- The ASSIGNED attribute should be OFF.
- The RESILIENT attribute should be OFF.
- The SUBTYPE attribute can be either STATIC or DYNAMIC.
- The TYPE attribute should not be PRINT.

For SUBTYPE STATIC with DEFAULT ON, Telserv starts presenting the service selection menu after all existing windows under the service are exhausted. Subsequent connections are mapped to dynamic windows.

`DISPLAY`

controls the display of service names in the Service Menu and controls the amount of SERVICE information provided on #ZSPI. This feature is provided for SERVICEs that are security-sensitive.

`LIB`

identifies a local library file name to be used to launch the *`file-name`* identified in PROGRAM. If not specified, the program is launched without a library. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4. Invalid or inaccessible file names are rejected with Telserv error 11.

`OWNER`

identifies the user name (group.user) that owns the service. When this attribute is specified, a client requesting a service is authenticated and access to the service is enforced by the Login Server. Invalid user names are rejected with Telserv error 1.

`PARAM`

contains a run-time parameter string up to 128 bytes to be passed in the STARTUP message of the program being launched. You can specify any string except N/A.If PROGRAM is not specified, this attribute is invalid and is rejected with Telserv error 4. You can reset this attribute by setting it empty quotation marks ("").

`PRI`

specifies the program's priority. Otherwise, Telserv uses its own priority minus one. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4.

`PROGRAM`

contains the local file name of the initial program to be launched by the Login Server when a session is established. Invalid or inaccessible file names are rejected with Telserv error 11. Invalid file codes (file code does not equal 100) are rejected with Telserv error 12.

`RESILIENT`

allows a Telserv client to recover from session failures. It is not supported on a DYNAMIC service and is rejected with Telserv error 4. There can be one STATIC window per resilient service.

`SUBTYPE`

specifies the service subtype.

`DYNAMIC`

defines a service that is not associated with a configures window, for example, a window that was added. A dynamic service automatically creates a window when a session is established.

`STATIC`

allows one or more windows to be added to a given service.

`SWAP`

identifies a local volume name to use as the swap file for the PROGRAM *`file-name`*. If not specified, the Login Server uses either the swap file specified in a previous ADD SERVICE command or, by default, a swap file determined by the system. This attribute is invalid if PROGRAM is not specified and is rejected with Telserv error 4. Invalid or inaccessible disc names are rejected with Telserv error 11.

`TYPE`

specifies the class of service.

`BLOCK`

configures a T6530 block mode service and initiates the session in block mode. If this attribute is selected, SETMODE 28 resets the terminal to block mode.

`CONVERSATION`

configures a conversational terminal service and initiates the session in conversation mode.

`PRINT`

configures a printer (write-only device) service.

ALTER WINDOW Command

The *`attribute-specs`* for the WINDOW object are:

```
[ CALLER "IP4-address"                          ]
[ CALLER6 "IP6-address"                         ]
[ ENDOFFILE 0 through 255 ]
[ ENDOFLINE 0 through 255 ]
[ ERASE 0 through 255     ]
[ INTERRUPT 0 through 255 ]
[ LINEKILL 0 through 255  ]
```

`CALLER`

allows a client to connect directly to a preconfigured STATIC window, based on the IPv4 address of the client. The CALLER attribute is intended to be used in conjunction with the SERVICE attribute, ASSIGNED. The CALLER attribute is required if the corresponding SERVICE object, to which the window belongs, has its ASSIGNED attribute ON. Otherwise, it is optional. Refer to WINDOW object in Considerations for the ADD Command for more information.

`CALLER6`

is a quoted IPv6 address of the Telnet client. This address is automatically assigned to this configured window upon connection. The IP address is specified in eight 16-bit hexadecimal numbers. For example:

```
"5f1c:ce00:df3f:b210:0030:0700:306a:e3e3"
```

Leading blanks in quoted string are invalid. If you specify leading blanks, Telserv will return an error 1.

`ENDOFFILE`

is a 1-byte ASCII value used to signify an end-of-input condition to a reader. The default value is CTRL-Y (%H19 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`ENDOFLINE`

is a 1-byte ASCII value used to signify that an input line is complete. The default value is a CARRIAGE-RETURN (%H0C in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`ERASE`

is a 1-byte ASCII value used to erase a single character of input. The default value is a BACKSPACE (%H08 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

`INTERRUPT`

is a 1-byte ASCII value used to send a break message to the process that owns the break on the window. The default value is a CTRL-C (%H03 in hexadecimal). This attribute is valid only when SERVICETYPE is set to block mode or conversation mode.

`LINEKILL`

is a 1-byte ASCII value used to delete the line being entered. The default value is CTRL-X (%H18 in hexadecimal). This attribute is valid only when SERVICETYPE is set to conversation mode.

Considerations for the ALTER Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

All Objects

Use the INFO command to view the current attributes.

PROCESS Object

- ALTER PROCES is a sensitive command.
- The value of MAXTERMINALS cannot be changed to a value lower than the current number of windows in session.

SERVICE Object

- All subordinate windows must be in the STOPPED summary state before SERVICE attributes can be altered.
- The SUBTYPE attribute cannot be altered from STATIC to DYNAMIC if subordinate windows exist.
- The ALTER command does not support the SERVICETYPE attribute; a window has to be STOPPED, DELETED, and ADDED with new attributes.

- Use the INFO SERVICE command with the SUB ALL syntax to view all windows subordinate to a service.
- Use the STOP WINDOW command to stop each window subordinate to a service.

WINDOW Object

- ALTER WINDOW is a sensitive command.
- The WINDOW object must be in the STOPPED summary state before its attributes can be altered. When the command is completed, the object remains in the same summary state that existed before the command was executed.
- Use the STATUS WINDOW command with the * syntax to view the summary state of all the windows.
- Use the STOP WINDOW command with the * syntax to put all the windows into the STOPPED summary state.
- The CALLER attribute requires a quoted IP address as its argument. Any leading blanks in the quoted IP address result in Telserv error 1. The CALLER attribute is required if the corresponding SERVICE object, to which the window belongs, has its ASSIGNED attribute ON. Otherwise, it is optional.

  Generally, multiple STATIC windows can not be configured using the same CALLER address. Note the following:

  - Multiple STATIC windows belonging to the same SERVICE having the ASSIGNED attribute set ON can have the same CALLER address.
  - STATIC windows belonging to different services having the ASSIGNED attribute set ON can not have the same CALLER address.
  - STATIC windows belonging to services having the ASSIGNED attribute set OFF can have the same CALLER address. In this case, the CALLER address is ignored. Subsequent use of the ALTER command to set an ASSIGNED attribute ON can result in an error if the CALLER address of any of the windows is a duplicate of a window for some other service having its ASSIGNED attribute set ON.
- STATIC windows must be in the STOPPED summary state before the CALLER address can be altered. Otherwise, an error occurs.
- The CALLER address for WINDOW objects should be defined before any attempt to alter the ASSIGNED attribute of a SERVICE of SUBTYPE STATIC.

Examples of the ALTER Command

The examples are designed to show correct ways of specifying command syntax.

SERVICE Object

```
ALTER SERVICE $znt.payroll, type conversation, display on
```

WINDOW Object

The following command sets the upper limit to 90 windows:

```
ALTER PROCESS $ZTNT, MAXTERMINALS 90
```

DELETE Command

The DELETE command removes a SERVICE or WINDOW from the subsystem.

The DELETE SERVICE command removes an existing SERVICE object. If SUB ALL is specified, the command will delete the specified SERVICE and all subordinate windows. All subordinate windows must first be in the STOPPED summary state. Issue an INFO SERVICE command with the SUB ALL option to obtain a listing of the subordinate window objects. Then issue a STOP WINDOW command to put each window in the STOPPED summary state. You can use an object-name template with this command.

The DELETE WINDOW command removes an existing WINDOW object. Before you can delete a window, it must be in the STOPPED summary state. Use the STOP WINDOW command to put the window in the STOPPED summary state.

DELETE is a sensitive command.

```
DELETE [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The DELETE command has the following *object-spec*:

```
object-type          object-name

SERVICE              service-name
WINDOW               window-name
```

DELETE SERVICE Command

The syntax for the DELETE SERVICE command is:

```
DELETE [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

DELETE WINDOW Command

The syntax for the DELETE WINDOW command is:

```
DELETE [ / OUT file-spec / ] WINDOW window-name
```

Considerations for the DELETE Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

All Objects

- DELETE is a sensitive command.
- When the DELETE operation finishes, any windows you specified for deletion are removed from the subsystem. This command reverses the effect of the ADD command.

Examples of the DELETE Command

The examples are designed to show correct ways of specifying command syntax.

SERVICE Object

```
DELETE SERVICE $ztnt.payroll
```

WINDOW Object

The following command deletes all windows under the $ZTNT process:

```
DELETE WINDOW $ZTNT.*
```

The next command deletes the window #WIN03 from process $ZTNT.

```
DELETE WINDOW $ZTNT.#WIN03
```

INFO Command

The INFO command returns information for the specified object. The INFO PROCESS command displays the current attribute settings for the Telserv subsystem. The INFO SERVICE command returns information about the SERVICE object when it was established. The INFO WINDOW command displays the current attribute settings for the specified window. Asterisks indicate which attributes have values that you can modify using the ALTER command. You can use an object-name template to specify windows with this command.

```
INFO [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

The INFO command has the following *object-spec*:

object-type	*object-name*
PROCESS	*process-name*
SERVICE	*service-name*
WINDOW	*window-name*

INFO PROCESS Command

The syntax for the INFO PROCESS command is:

```
INFO [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

The display for the INFO PROCESS command without the DETAIL option is:

```
TELSERV Info PROCESS

 *Max Terminals     Total Terminals     *Timeout Value      Port
 256                0                   3                   9876
```

`Max Terminals`

is the maximum number of terminals that can be present in the system.

`Total Terminals`

is the actual number of terminals.

`Timeout Value`

is the number of minutes after which inactive terminals are disconnected. If inactive terminals are not to be disconnected, this value is displayed as N/A.

`Port`

is the local TCP port number this process is using.

The display for the INFO PROCESS command with the DETAIL option is:

```
TELSERV Detailed Info PROCESS \SYS1.$MTNT

 PCPU................... 11            BCPU................... N/A
 PPIN................... 379           BPIN................... N/A
 TACL................... ON            Transport Process...... $ZTC0
*Menu................... ON            Transport Type......... TCP/IP
*Timeout Value.......... N/A           Port................... 4040
*Banner Timeout Value... N/A           Total Services......... 2
*Max Terminals.......... 256           Total Terminals........ 2
 Program................ . \SYS1.$DISK2.MYTWD21.TELSERV
*CPU List................. 0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13
,14 ,15, ,16
*DROPCR.................. ON
```

`PCPU`

is the primary process CPU.

`BCPU`

is the backup process CPU.

`PPIN`

is the primary process PIN.

`BPIN`

is the backup process PIN.

`TACL`

ON indicates that TACL should be one of the services listed in the service menu at the beginning of an incoming session.

`Transport Process`

is the name of the transport process in use.

`Menu`

is the service menu. ON indicates the service menu should be displayed at the beginning of an incoming session. OFF indicates the service menu should not be displayed at the beginning of an incoming session.

`Transport Type`

indicates the type of transport process, either TCP/IP or IPX/SPX.

`Timeout Value`

is the number of minutes after which inactive terminals are disconnected. If inactive terminals are not to be disconnected, this value is displayed as N/A.

`Port`

is the local TCP port number this process is using.

`Banner Timeout Value`

is the time (in minutes) after which a terminal can remain inactive following presentation of the banner in the dynamic window. If this value is reached, any inactive terminal is disconnected. If inactive terminals are not to be disconnected, this value is displayed as N/A.

`Total Services`

is the number of services.

`Max Terminals`

is the maximum number of terminals that can be present in the system.

`Total Terminals`

is the number of terminals.

`Program`

is the name of the program that this process was started from.

`CPU List`

is the list of processors on which the application can be launched when no CPU attribute is specified for the SERVICE object and the $CMON process is not available on the system.

`DROPCR`

indicates that Telserv does not provide a null response for embedded termination characters while replying to [WRITE]READ[X] calls.

INFO SERVICE Command

The syntax for the INFO SERVICE command is:

```
INFO [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

The display for the INFO SERVICE command without the DETAIL option is:

```
TELSERV Info SERVICE

Name        *Type          *Subtype    *Display   *Program
MYSERV      BLOCK          STATIC      ON         N/A
```

Note. N/A is displayed for any field that does not have a default value and for which you did not specify a value.

`Name`

is the name of the SERVICE object.

`Type`

is the class of service: `BLOCK`, `CONVERSATION`, or `PRINT`.

`SubType`

is the service subtype: `DYNAMIC` or `STATIC`.

`Display`

indicates whether or not service names are displayed in the Service Menu. ON indicates that service names should be displayed in the Service Menu at the beginning of an incoming session. OFF indicates that service names should not be displayed in the Service Menu at the beginning of an incoming session.

`Program`

is the local file name of the initial program to be launched by the Login Server when a session is established.

The display for the INFO SERVICE command with the DETAIL option is:

```
TELSERV Detailed Info SERVICE \COMM.$MTNT.MYSERV

*Type................... BLOCK         *Subtype................ STATIC
*Display................ ON            *Autodelete............. OFF
*Owner.................. N/A           *Access................. ALL
*CPU.................... N/A           *Pri.................... N/A
*Swap................... N/A
*Program................ N/A
*Lib.................... N/A
*Resilient.............. OFF
*Param.................. N/A
```

Note. N/A is displayed for any field that does not have a default value and for which you did not specify a value.

`Type`

is the class of service: `BLOCK`, `CONVERSATION`, or `PRINT`.

`Subtype`

is the service subtype: `DYNAMIC` or `STATIC`.

`Display`

indicates whether or not service names are displayed in the Service Menu. ON indicates that service names should be displayed in the Service Menu at the beginning of an incoming session. OFF indicates that service names should not be displayed in the Service Menu at the beginning of an incoming session.

`Autodelete`

indicates when the service is deleted. ON indicates the service is to be deleted when all WINDOWS associated with that service are deleted. OFF indicates the service is NOT to be deleted when all WINDOWS associated with that service are deleted.

`Owner`

is the user name that owns the service.

`Access`

is the authorized access for this service.

`CPU`

is the CPU to use to NEWPROCESS the program identified in the Program field.

`Pri`

specifies the program's priority.

`Swap`

is a local volume name used for the Program swap file.

`Program`

is the local file name of the initial program to be launched by the Login Server when a session is established.

`Lib`

is a local library file name to be used when launching the program identified in the Program field.

`Resilient`

indicates if the TELNET client will recover from failures. ON indicates the TELNET client is to recover from session failures. OFF indicates the TELNET client is not to recover from session failures.

`Param`

is the run-time parameter string which is to be passed in the STARTUP message to the program being launched.

INFO WINDOW Command

The syntax for the INFO WINDOW command is:

```
INFO [ / OUT file-spec / ] WINDOW window-name [ , DETAIL ]
```

The display for the INFO WINDOW command without the DETAIL option is:

```
Window Name   Service Name   Service Type  Local Port   Foreign IP Address
#PTY0007      TACL           CONVERSATION  N/A          N/A
```

`Window Name`

is the name of the window.

`Service Name`

is the name of the service associated with this terminal.

`Service Type`

is the type of the service associated with this window.

`Local Port`

is the local TCP port number for this connection. This field is valid only when TCP is the transport layer used.

`Foreign IP Address`

is the IP address of the remote machine for this connection. This field is valid only when TCP is the transport layer used.

The display for the INFO WINDOW command with the DETAIL option is:

```
TELSERV Detailed Info WINDOW \STUMPY.#PTY0002

*Erase.................. %H08          *Kill-Line.............. %H18
*End-of-File............ %H19          *End-of-Line............ %H0D
*Interrupt.............. %H03
Service Type........... CONVERSATION   Service Name........... TACL
Local Port............. 9876           Foreign Port........... 1200
Local IP Address....... 130.252.45.98
Foreign IP Address..... 130.252.36.114
Netware Local Address.. N/A
Netware Remote Address. N/A
```

```
TELSERV Detailed Info WINDOW \STUMPY.#PTY0007

*Erase.................. %H08          *Kill-Line.............. %H18
*End-of-File............ %H19          *End-of-Line............ %H0D
*Interrupt.............. %H03
 Service Type........... CONVERSATION   Service Name........... TACL
 Local Port............. N/A            Foreign Port........... N/A
 Local IP Address....... N/A
 Foreign IP Address..... N/A
 Netware Local Address.. 0000001A:08008E002B00:4029
 Netware Remote Address. 0000001A:0020AF334DCA:7000
```

In the first display, TCP is the transport layer used; in the second display, SPX is the transport layer used.

`Erase`

is the one-byte character value used to erase a single character of input.

`Kill-Line`

is the one-byte character value used to delete the line currently being entered.

`End-of-File`

is the one-byte character value used to signify the end of the input condition.

`End-of-Line`

is the one-byte character value used to signify that an input line is complete.

`Interrupt`

is the one-byte character value used to send a break message to the process that owns the break on the terminal.

`Service Type`

is the type of the service associated with this window.

`Service Name`

is the name of the service associated with this terminal.

`Local Port`

is the local TCP port number for this connection. This field is valid only when TCP is the transport layer used.

`Foreign Port`

is the remote TCP port number for this connection. This field is valid only when TCP is the transport layer used.

`Local IP Address`

is the local IP address associated with this connection. This field is valid only when TCP is the transport layer used.

`Foreign IP Address`

is the IP address of the remote machine for this connection. This field is valid only when TCP is the transport layer used.

`Netware Local Address`

is the local Netware address associated with this connection. This field is valid only when SPX is the transport layer used. The format of the SPX address is *xxxxxxxx:yyyyyyyyyyyy:zzzz*.

xxxxxxxx

is the IPX network address.

yyyyyyyyyyyy

is the MAC address of the node.

zzzz

is the socket number.

`Netware Remote Address`

is the remote Netware address associated with this connection. This field is valid only when SPX is the transport layer used. The format of this value is the same as that for `Netware Local Address`.

Considerations for the INFO Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

PROCESS Object

- The INFO PROCESS command does not report the summary state of the process.
- The STATUS PROCESS and STATS PROCESS commands provide information about the summary state and statistics of the PROCESS respectively.

WINDOW Object

- The INFO WINDOW command does not report the summary state of the window.
- The STATUS WINDOW and STATS WINDOW command provide information about summary state and the statistics of WINDOW, respectively.

Examples of the INFO Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

The following command displays non-detailed information for the assumed process.

```
INFO PROCESS
```

The following command displays detailed information for the $ZTNT process.

```
INFO PROCESS $ZTNT, DETAIL
```

SERVICE Object

```
INFO SERVICE $znt.payroll, detail
```

WINDOW Object

```
INFO WINDOW $ztnt.#win25, detail
```

LISTOPENS Command

The LISTOPENS PROCESS command displays information about the openers of the Telserv server process. The LISTOPENS WINDOW command identifies applications that have opened the specified window.

```
LISTOPENS [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The LISTOPENS command has the following *object-spec*:

```
object-type          object-name

PROCESS              process-name
WINDOW               window-name
```

LISTOPENS PROCESS Command

The syntax for the LISTOPENS PROCESS command is:

```
LISTOPENS [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The display for the LISTOPENS PROCESS command is:

```
TELSERV Listopens PROCESS \COMM.$MTNT

Window    Openers              PCPU PPIN PLFN BCPU BPIN BLFN   Service
#ZSPI     \COMM    $ZNET       2    239  4    N/A  N/A  N/A    ZSPI
```

`Window`

is the window name that the opener opened.

`Openers`

is the process name of openers of the Telserv process.

`PCPU`

is the primary CPU of the opener process.

`PPIN`

is the primary PIN of the opener process.

`PLFN`

is the logical file number of the primary opener process.

`BCPU`

is the backup CPU of the opener process.

`BPIN`

is the backup PIN of the opener process.

`BLFN`

is the logical file number of the backup opener process.

`Service`

is the service name of the window.

LISTOPENS WINDOW Command

The syntax for the LISTOPENS WINDOW command is:

```
LISTOPENS [ / OUT file-spec / ] WINDOW window-name
```

The display for the LISTOPENS WINDOW command is:

```
Window      Opener              PCPU PPIN PLFN BCPU BPIN BLFN  Service
#WIN        N/A                 N/A  N/A  N/A  N/A  N/A  N/A   N/A
#ZSPI       \COMM    $ZNET      2    239  4    N/A  N/A  N/A   ZSPI
```

`Window`

is the window name that the opener opened.

`Opener`

is the process name of openers of the Telserv process.

`PCPU`

is the primary CPU of the opener process.

`PPIN`

is the primary PIN of the opener process.

`PLFN`

is the logical file number of the primary opener process.

`BCPU`

is the backup CPU of the opener process.

`BPIN`

is the backup PIN of the opener process.

`BLFN`

is the logical file number of the backup opener process.

`Service`

is the service name of the window.

Examples of the LISTOPENS Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

```
LISTOPENS PROCESS $ztnt
```

The following command shows information about the openers of the assumed process:

```
LISTOPENS PROCESS
```

The following command shows information about the openers of the $ZTNT process:

```
LISTOPENS PROCESS $ZTNT
```

WINDOW Object

```
LISTOPENS WINDOW $ztnt.#win25
```

NAMES Command

The NAMES command lists the names and types of all, or a subset of, the objects known to the subsystem. You can use an object-name template with this command.

```
NAMES [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The NAMES command has the following *object-spec*:

```
object-type          object-name

PROCESS              process-name
SERVICE              service-name
WINDOW               window-name
null                 process-name
```

NAMES PROCESS Command

The syntax for the NAMES PROCESS command is (SUB ALL is the default value):

```
NAMES [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The display for the NAMES PROCESS command is:

```
TELSERV Names PROCESS \COMM.$MTNT

PROCESS
$MTNT

SERVICE
WINSERV  TACL     ZVTL     ZTELNET  ZBLOCK   ZCONV    ZPRINT   ZSPI

WINDOW
#WIN
```

PROCESS

lists the PROCESS objects.

SERVICE

lists the SERVICE objects.

WINDOW

lists the WINDOW objects.

NAMES SERVICE Command

The syntax for the NAMES SERVICE command is:

```
NAMES [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

The display for the NAMES SERVICE command is:

```
TELSERV Names SERVICE \COMM.WINSERV

SERVICE
WINSERV  TACL     ZVTL     ZTELNET  ZBLOCK   ZCONV    ZPRINT   ZSPI

WINDOW
#WIN
```

`SERVICE`

lists the SERVICE objects.

`WINDOW`

lists the WINDOW objects.

NAMES WINDOW Command

The syntax for the NAMES WINDOW command is:

```
NAMES [ / OUT file-spec / ] WINDOW window-name
```

The display for the NAMES WINDOW command is:

```
TELSERV Names WINDOW \COMM.#WIN

WINDOW
#WIN
```

`WINDOW`

lists the WINDOW objects.

NAMES *null* Command

The syntax for the NAMES `null` command is:

```
NAMES [ / OUT file-spec / ] process-name
```

The display for the NAMES *null* command is:

```
TELSERV Names PROCESS \COMM.$MTNT

PROCESS
$MTNT

SERVICE
WINSERV  TACL     ZVTL     ZTELNET  ZBLOCK   ZCONV    ZPRINT   ZSPI

WINDOW
#WIN
```

`PROCESS`

lists the PROCESS objects.

`SERVICE`

lists the SERVICE objects.

`WINDOW`

lists the WINDOW objects.

Examples of the NAMES Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

```
NAMES PROCESS $ZTNT
```

SERVICE Object

```
NAMES SERVICE $znt.*
```

WINDOW Object

The following command displays names of all windows under the assumed process:

```
NAMES WINDOW *
```

The next command displays names of all windows under the $ZTNT process.

```
NAMES WINDOW $ZTNT.*
```

START WINDOW Command

The START WINDOW command initiates the operation of a window and places the window in the STARTED summary state. You can use an object-name template with this command.

```
START [ / OUT file-spec / ] WINDOW window-name
```

Considerations for the START WINDOW Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

- The START command is a sensitive command.
- The START command will complete successfully only if the object is in the STOPPED summary state.
- Use the STOP or ABORT command to terminate the operation of windows.

Examples of the START WINDOW Command

The examples are designed to show correct ways of specifying command syntax.

The following command starts all windows under the assumed process.

```
START WINDOW *
```

The command below starts the window #WN1 under the assumed process.

```
START WINDOW #WN1
```

The next command starts the window #WI34 under the $ZTNT process.

```
START WINDOW $ZTNT.#WI34
```

STATS Command

The STATS command displays statistical information for the specified object. The time stamp for the reset is recorded. You can use an object-name template to specify windows with this command.

```
STATS [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ] [ , RESET ]
```

The STATS command has the following *object-spec*:

object-type	*object-name*
PROCESS	*process-name*
SERVICE	*service-name*
WINDOW	*window-name*

STATS PROCESS Command

The syntax for the STATS PROCESS command is:

```
STATS [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , RESET ]
```

The display for the STATS PROCESS command is:

```
TELSERV Stats PROCESS \COMM.$MTNT

 Sample Time............ 22 Jul 1994, 11:39:55.980
 Reset Time............. 22 Jul 1994, 11:08:36.615

 In Connections......... 0              Average Active Terms... 1
 Total Opens............ 1              Total Spi Requests..... 273
 Total Net Data......... 0              Total User Data........ 0
 Total Net Requests..... 0              Total User Requests.... 273
```

When RESET is specified for PROCESS objects, all the counters are reset to 0.

`Sample Time`

is the time when the statistics were taken.

`Reset Time`

is the time when the statistics were reset.

`In Connections`

is the number of incoming Telserv connection requests.

`Average Active Terms`

is the average number of windows active in the system in the last five minutes.

`Total Opens`

is the total number of open requests processed.

`Total Spi Requests`

is the number of SPI requests processed.

`Total Net Data`

is the total number of bytes received from the network.

`Total User Data`

is the total number of bytes sent by the user to the network.

`Total Net Requests`

is the total number of packets received from the network.

`Total User Requests`

is the total number of user requests.

STATS SERVICE Command

The syntax for the STATS SERVICE command is:

```
STATS [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , RESET ]
```

The display for the STATS SERVICE command is:

```
TELSERV Stats SERVICE \COMM.MYSERV

 Sample time............ 22 Jul 1994, 11:40:08.511
 Reset time............. 22 Jul 1994, 11:39:57.524
 Sessions............... 0              Windows................ 0
 Aborts................. 0              Inuses................. 0
```

When RESET is specified for SERVICE objects, the Sessions and Aborts counters are reset to 0.

`Sample time`

is the time when the statistics were displayed.

`Reset time`

is the time when the statistics were reset.

`Sessions`

is the number of connection attempts to the service.

`Windows`

is the number of windows associated with a service. The number of configured windows cannot be reset with the RESET option.

`Aborts`

is the number of connections aborted because the client could not be authenticated, the client did not have sufficient privileges to access the service, or the login process was not able to launch the initial program.

`Inuses`

is the maximum number of windows in use at sample time. The number of windows in use cannot be reset with the RESET option.

STATS WINDOW Command

The syntax for the STATS WINDOW command is:

```
STATS [ / OUT file-spec / ] WINDOW window-name [ , RESET ]
```

The display for the STATS WINDOW command is:

```
TELSERV Stats WINDOW \COMM.#WIN

 Sample-Time............ 22 Jul 1994, 11:40:09.535
 Reset-Time............. 22 Jul 1994, 11:40:08.666

 Name................... #WIN
 User Bytes............. 0              Net Bytes.............. 0
 Read Requests.......... 0              Write Requests......... 0
 Writeread Reqs......... 0              Control Reqs........... 0
 Setmode Reqs........... 0              Cancel Reqs............ 0
```

When RESET is specified for WINDOW objects, all the counters are reset to 0.

`Sample Time`

is the time when the statistics were taken.

`Reset Time`

is the time when the statistics were reset.

`Name`

is the name of the WINDOW object.

`User Bytes`

are the number of bytes sent by the user.

`Net Bytes`

are the number of bytes received from the network.

`Read Requests`

are the number of read requests posted by the user.

`Write Requests`

are the number of write requests posted by the user.

`Writeread Reqs`

are the number of writeread requests posted by the user.

`Control Reqs`

are the number of control requests posted by the user.

`Setmode Reqs`

are the number of SETMODE requests performed by the user.

`Cancel Reqs`

are the number of requests that were canceled before completion by the user.

Considerations for the STATS Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

All Objects

- Use the STATS command with the RESET option to initialize the statistical counters.
- STATS is a nonsensitive command only when used with the RESET option.

WINDOW Object

- The summary state of the object does not prevent the successful completion of the STATS command.

Examples of the STATS Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

```
STATS PROCESS $ztnt
```

SERVICE Object

```
STATS SERVICE $znt.payroll
```

WINDOW Object

The following command displays statistics for all windows under the assumed process.

```
STATS WINDOW *
```

The following command displays statistics for the assumed process and then resets the counters to zero.

```
STATS PROCESS, RESET
```

The next command displays statistics for the window #WN1 under the assumed process.

```
STATS WINDOW #WN1
```

STATUS Command

The STATUS command displays the status of the specified object. You can use an object-name template the STATUS WINDOW command.

```
STATUS [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

The STATUS command has the following *object-spec*:

```
object-type          object-name

PROCESS              process-name
WINDOW               window-name
```

STATUS PROCESS Command

The syntax for the STATUS PROCESS command is:

```
STATUS [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

The display for the STATUS PROCESS command without the DETAIL option is:

```
TELNET Status PROCESS \SYSA.$XTNT

Status:  STARTED
```

`Status`

is the current summary state of the process. The process should always be in the STARTED summary state.

The display for the STATUS PROCESS command with the DETAIL option is:

```
TELNET Detail Status PROCESS \SYSA.$XTNT

Status ...............STARTED
```

`Status`

is the current summary state of the process. The process should always be in the STARTED summary state.

STATUS WINDOW Command

The syntax for the STATUS WINDOW command is:

```
STATUS [ / OUT file-spec / ] WINDOW window-name [ , DETAIL ]
```

The display for the STATUS WINDOW command without the DETAIL option is:

```
TELSERV Status WINDOW

     Name                Status                  Data Transmission
     #PTY0007            STARTED                 8BITS
```

`Name`

is the name of the WINDOW object.

`Status`

is the summary state of the object.

`Data Transmission`

is the number of bits used in transmission.

The display for the STATUS WINDOW command with the DETAIL option is:

```
TELSERV Detailed Status WINDOW \STUMPY.$ZTN1.#PTY0007

 Name................... #PTY0007       Status................. STARTED
 Data Transmission...... 8BITS
```

`Name`

is the name of the window.

`Status`

is the current summary state of the window.

`Data Transmission`

is the number of bits used in transmission.

Considerations for the STATUS Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

All Objects

- The command completes regardless of the summary state of the objects.

- The STATUS command reports on, but does not change, the summary state of specified objects.
- STATUS is a nonsensitive command; any valid user on the system can issue this command.

PROCESS Object

- Use the STATUS PROCESS with the SUB ALL syntax to display the status of all the WINDOWS in the subsystem.
- Use the INFO PROCESS command to view the current attribute settings.

Examples of the STATUS Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

```
STATUS PROCESS $ztnt
```

WINDOW Object

The following command reports the status on all windows under the assumed process.

```
STATUS WINDOW *
```

The next command reports the status of the $ZTNT process.

```
STATUS PROCESS $ZTNT
```

STOP Command

The STOP PROCESS command terminates the activity of the specified Telserv process in a normal, orderly manner. The STOP WINDOW command terminates the operation of the specified window, leaving them in the STOPPED summary state. If you attempt to stop a window that is in use, the operation returns an error and does not complete. You can use an object-name template with this command. This is a sensitive command.

```
STOP [ / OUT file-spec / ] [ object-spec ]
```

The STOP command has the following *object-spec*:

```
object-type          object-name

PROCESS              process-name
WINDOW               window-name
```

STOP PROCESS Command

The syntax for the STOP PROCESS command is:

```
STOP [ / OUT file-spec / ] PROCESS [ process-name ]
```

STOP WINDOW Command

The syntax for the STOP WINDOW command is:

```
STOP [ / OUT file-spec / ] WINDOW [ window-name ]
```

Considerations for the STOP Command

The considerations contain information about command use that should be read before studying the command syntax in detail.

All Objects

STOP is a sensitive command.

PROCESS Object

To terminate the Telserv process immediately, use the ABORT command.

Examples of the STOP WINDOW Command

The examples are designed to show correct ways of specifying command syntax.

```
STOP WINDOW $ztnt.#win*
```

The following command terminates the operation of all windows under the assumed process.

```
STOP WINDOW *
```

The next command terminates the operation of all windows that have names beginning with #WI and that are under the $ZTNT process.

```
STOP WINDOW $ZTNT.#WI*
```

TRACE Command

Use the TRACE command to collect trace information for a PROCESS or WINDOW object. Issue the TRACE command (without the STOP option and with the TO option) to specify trace options and start tracing. While tracing is on, issue the TRACE command (without the STOP or TO options) to modify COUNT, RECSIZE, or any of the SELECT specifications. Issue the TRACE command with the STOP option to stop tracing. TRACE is a sensitive command.

To examine trace files, use the formatter available with the PTrace program. The PTrace program is described in Section 6, Using PTrace.

Caution. The trace operation can significantly increase CPU use by the Telserv process. To avoid problems with other processes in the CPU, you should lower the priority of the Telserv process before you issue the TRACE command.

```
TRACE [ / OUT file-spec / ] [ object-spec ]

   { , STOP                    }
   { [ , COUNT count        ]
     [ , NOCOLL             ]
     [ , PAGES pages        ]
     [ , RECSIZE size       ]
     [ , SELECT select-spec ]
     [ , TO file-spec       ]             ***
     [ , WRAP               ] }


    *** This attribute is required when a trace is started.
```

The TRACE command has the following *object-spec*:

```
object-type          object-name

PROCESS              process-name
WINDOW               window-name
```

`file-spec`

causes any SCF output generated for this command to be directed to the specified file.

`object-spec`

designates the objects to be traced. If object-spec is omitted, the assumed object is traced.

`STOP`

ends the trace operation. A TRACE command must include either the STOP option or the TO option.

`COUNT`

specifies the number of trace records to be captured. *count* is an integer in the range -1 through 32767. If you omit this option or if *count* equals -1, records are accumulated until you use the STOP option.

`NOCOLL`

prevents the initiation of the trace collector process.

`PAGES`

designates how much space, in units of pages, is allocated in the extended data segment used for tracing. You can specify PAGES only when initiating a trace, not when you are modifying its parameters. *`pages`* is an integer in the range 4 through 64 or equal to 0. If you omit this option or specify 0, the default value 64 is applied to the trace.

`RECSIZE`

specifies the length of the data in the trace data records. *`size`* is an integer in the range 16 through 4050 or equal to 0. The length of the trace header, which is eight bytes, is not included in *`size`*. If you omit this option or specify 0, the default value of 120 bytes is applied to the trace.

`SELECT`

selects which records are collected and written to the trace file. Records are identified by trace record type, using the descriptions that appear in PTrace display formats. The *`select-spec`* is one or more of the following specifications:

```
{ keyword                       }
{ ( keyword [ , keyword ] ...) }

{ number                      }
{ ( number [ , number ] ...) }
```

where *`keyword`* or *`number`* is one or more of the following values:

For PROCESS objects:

```
[ ALL    ] [ -1 ]
[ CONN   ] [  2 ]
[ LOGIN  ] [  9 ]
[ ONLINE ] [ 14 ]
[ OPEN   ] [  4 ]
[ RECV   ] [ 10 ]
[ SCB    ] [  8 ]
[ SPI    ] [ 13 ]
[ TELOPT ] [ 15 ]
```

For WINDOW objects:

```
[ ALL    ] [ -1 ]
[ ONLINE ] [ 22 ]
[ RCB    ] [ 21 ]
[ SOCK   ] [ 18 ]
[ TTY    ] [ 20 ]
```

`ALL ( or blank)`

Specifies that all trace options are selected. This is the default value. The trace file can get very large very quickly. It will wrap, if requested, or will simply stop tracing when the TO file gets full.

`CONN`

Traces connection activity between the client and Telserv.

`LOGIN`

Trace user login process.

`ONLINE`

This keyword causes trace records to be displayed on the Telserv home terminal if online trace is enabled.

`OPEN`

Trace Open Control Block and Process Open Control Block.

`RECV`

Trace system message, user message and reply.

`SCB`

Trace Service Control Block.

`SPI`

Trace SPI command.

`TELOPT`

Trace the TELNET negotiation option.

`RCB`

Trace TTY Request Control Block.

`SOCK`

Trace SOCKET IN, SOCKET OUT data.

`TTY`

Trace Terminal Control Block.

`TO`

specifies the name of the file the results of the trace operation are to be placed in. TO is a required attribute when starting a trace.

Examples of the TRACE Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

The following command traces the PROCESS object $ZTNT, writes results into the file named TLW, prevents the trace collector process from being initiated, and selects the tracing of all Telserv process activity.

```
TRACE PROCESS $ZTNT, TO $SYSA.TRACES.TLW, NOCOLL, SELECT ALL
```

WINDOW Object

The following command traces the $ZTNT.#WI2 window, writes the results into the file named TELW, allows the trace collector process to be initiated, and requests the tracing of Telserv negotiations.

```
TRACE WINDOW $ZTNT.#WI2, TO $SYSA.TRACES.TELW, SELECT TELOPT
```

The next command ends the tracing of the window #WI2 under the assumed process.

```
TRACE WINDOW #WI2, STOP
```

VERSION Command

In response to the VERSION command, the Telserv subsystem initiates a display of its version number and banner (product name, product number, and RVU date).

```
VERSION [ / OUT file-spec / ] [ object-spec ] [ , DETAIL ]
```

The VERSION command has the following *object-spec*:

object-type	*object-name*
PROCESS	*process-name*
null	

VERSION PROCESS Command

The syntax for the VERSION PROCESS command is:

```
VERSION [ / OUT file-spec / ] PROCESS [ process-name ]

   [ , DETAIL ]
```

The display for the VERSION PROCESS command without the DETAIL option has the following format:

```
VERSION PROCESS \COMM.$MTNT: TELSERV - T9553D30 - (21JUL94) - (21JUL94)
```

The display for the VERSION PROCESS command with the DETAIL option has the following format:

```
Detailed VERSION PROCESS \COMM.$MTNT
  SYSTEM \COMM
    TELSERV - T9553D30 - (21JUL94) - (21JUL94)
    GUARDIAN - T9050 - (D30)
    SCF KERNEL - T9082D30 - (31OCT94) (15MAY94)
    TELSERV PM - T6245D30-31OCT94-22JUL94
```

VERSION ***null*** Command

The syntax for the VERSION *null* command is:

```
VERSION [ / OUT file-spec / ] [ process-name ] [ , DETAIL ]
```

The display for the VERSION *null* command has the following format:

```
VERSION PROCESS \COMM.$MTNT: TELSERV - T9553D30 - (21JUL94) - (21JUL94)
```

The display for the VERSION *null* command with the DETAIL option has the following format:

```
Detailed VERSION PROCESS \COMM.$MTNT
  SYSTEM \COMM
    TELSERV - T9553D30 - (21JUL94) - (21JUL94)
    GUARDIAN - T9050 - (D30)
    SCF KERNEL - T9082D30 - (31OCT94) (15MAY94)
    TELSERV PM - T6245D30-31OCT94-22JUL94
```

Examples of the VERSION Command

The examples are designed to show correct ways of specifying command syntax.

PROCESS Object

The following command displays the banner of the process $ZTNT.

```
VERSION PROCESS $ZTNT
```

null Object

The following command displays the banner of the assumed process:

```
VERSION
```

The following command displays the banner for the assumed process as well as the banners for Guardian, the SCF Kernel, and the product module.

```
VERSION, DETAIL
```

6 Using PTrace

This section contains the subsystem-specific details for using PTrace to format Telserv subsystem trace files created through the use of the SCF TRACE command. These details include the device type and subtype for the Telserv subsystem, any variations to the standard PTrace commands, applicable SELECT keywords, and special considerations for using PTrace.

For complete information on using PTrace, refer to the *PTrace Reference Manual*.

Device Type and Subtype

The device type and subtype are 4 and 6, respectively.

PTrace Commands

The Telserv subsystem supports the following PTrace commands:

```
HEX
OCTAL
SELECT
```

The HEX and OCTAL commands are implemented in the standard PTrace manner. The SELECT command varies slightly from the standard, as described in this section.

The Telserv subsystem does not support the following PTrace commands:

```
TEXT
EBCDIC
DETAIL
SETTRANSLATE
TRANSLATE
TEST
FILTER
```

SELECT Command

The SELECT command establishes the selection criteria that control which trace records are to be displayed.

```
SELECT [  mask                            ]
       [  keyword                         ]
       [  ( keyword [ , keyword ] ... )  ]
```

mask

is a decimal integer that specifies a selection mask. The number is converted into a 32-bit mask and saved as an enumerated value. The acceptable range is 0 through 65535.

keyword

is one of the keywords listed in Table 6-1.

Table 6-1. SELECT Keywords for the Telserv Subsystem

Keyword	Description
Keywords for the PROCESS object:	
ALL	Specifies that all trace options are selected. This is the default value. The trace file can get very large very quickly. It will wrap, if requested, or will simply stop tracing when the TO file gets full.
CONN	Trace connection activity between client and Telserv.
LOGIN	Trace user login process.
ONLINE	This keyword causes trace records to be displayed on the Telserv home terminal if online trace is enabled.
OPEN	Trace Open Control Block and Process Open Control Block.
RECV	Trace system message, user message, and reply.
SCB	Trace Service Control Block.
SPI	Trace SPI command.
TELOPT	Trace the TELNET negotiation option.
Keywords for the WINDOW object:	
ALL	Specifies that all trace options are selected. This is the default value. The trace file can get very large very quickly. It will wrap, if requested, or will simply stop tracing when the TO file gets full.
ONLINE	This keyword causes trace records to be displayed on the Telserv home terminal if online trace is enabled.
RCB	Trace TTY Request Control Block.
SOCK	Trace SOCKET IN, SOCKET OUT data.
TTY	Trace Terminal Control Block.

Part III. Accessing and Using Telserv from a Terminal

7 Accessing and Using Telserv Services

As part of the Telserv product, HP provides a TELNET client that runs on the NonStop system. Other vendors provide TELNET client software as part of their terminal emulator products.

The client features included in the HP NonStop TELNET client differ from the features other vendors provide in their terminal emulator products. While other vendor's products support the 6530 terminal type, the HP NonStop TELNET client runs in NVT mode only.

Telserv communicates with client applications by using the TELNET protocol. The user invokes a TELNET client application program that connects to the specified remote machine (see Using TELNET to Connect to Telserv on page 8-1). Telserv then displays the Welcome Banner and the Service menu, for example:

```
WELCOME To idev [PORT $ZTC5 #23 WINDOW $ZTN5.#PTT7HET]
TELSERV - T9553D40 - (10SEP00) - (IPMADD)

Available Service:

TACL    EXIT

Enter Choice>
```

In this example:

- `idev` is the HP NonStop host name.
- `$ZTC5` is the Transport process name.
- `23` is the port number.
- `$ZTN5` is the Telserv process name.
- `#PTT7HET` is the window name.

When users select a service, they might be prompted for a logon ID and password. The client software then passes the keystrokes from the user terminal to the remote host and displays the output from the host on the user terminal.

The TELNET Protocol

The TELNET protocol provides a bidirectional, byte-oriented communications facility.

The TCP/IP or IPX/SPX connection is used to transfer data and TELNET control functions are referred to as *`TELNET connections`*.

Note. The interface to IPX/SPX is available on G-series systems only.

Both ends of a connection, the TELNET client and Telserv, negotiate TELNET control functions once the connection has been established. Control functions, which allow either end of the connection to dynamically modify the connection characteristics, are divided into two categories: commands and options. A command causes some action to occur, whereas an option simply negotiates whether or not a command will be used.

For more information, see TELNET Client Commands and TELNET Commands on page 9-1.

Modes of Operation

If you use client software from another vendor, you can interactively select the modes of operation described below. Note that the TELNET client provided by HP does not allow you to negotiate modes of operation.

Line Mode and Character Mode

In character mode (the default), the TELNET client immediately sends each character you type to the remote system. In line mode, the text you type is echoed locally, then sent to the remote host when you press the RETURN key (start and stop bits indicate the beginning and end of the data string).

To select line mode, use the TELNET client mode command (see mode Command on page 9-3).

Note that a TELNET client enters the mode the user specifies only if the remote host can enter that mode.

ASCII and Binary Mode

Telserv supports both ASCII and binary transfer modes. ASCII is the default mode. When in ASCII mode, Telserv performs end-of-line processing of incoming and outgoing data.

As a terminal user, you can toggle between ASCII and binary modes by following these steps:

Note. These instructions assume that you are using the client software from Sun Microsystems.

1. Access Telserv.

2. From the Welcome banner screen, use ENTER CHOICE to select a service application, such as TACL

   ```
   WELCOME To idev [PORT $ZTC5 #23 WINDOW $ZTN5.#PTT7HET]
   TELSERV - T9553D40 - (10SEP00) - (IPMADD)

   Available Service:

   TACL    EXIT

   Enter Choice>
   ```

3. Log on to the NonStop system.
4. Press control escape; that is, press caret (^) and left angle bracket (]).
5. The TELNET prompt appears

   ```
   telnet>
   ```

6. To see a description of the toggle options, type the following TELNET client command:

   ```
   telnet> toggle ?
   ```

   Toggle option are described in toggle Command on page 9-6.

7. If you want a display of options as they are processed, issue the following command before issuing the command described in Step 8.

   ```
   telnet> toggle options
   ```

8. When your current session is in ASCII mode, switch to binary mode by issuing the following TELNET client command:

   ```
   telnet> toggle binary
   ```

9. If you have issued the `toggle options` command prior to issuing the `toggle binary` command, the following display appears:

   ```
   Negotiating binary mode with remote host

   SENT DO BINARY
   SENT WILL BINARY
   RCVD WILL BINARY
   RCVD DO BINARY
   ```

10. Press RETURN to go back to the service application prompt, for example:

    ```
    TACL>
    ```

11. To toggle back to ASCII mode, repeat steps 4, 5, 7, and 8. The following display appears:

    ```
    Negotiating binary mode with remote host

    SENT DONT BINARY
    SENT WONT BINARY
    RCVD WONT BINARY
    RCVD DONT BINARY
    ```

8 Using TELNET to Connect to Telserv

You can establish communications with Telserv from any system on the network that has a HP NonStop TELNET client. You invoke the TELNET client on your terminal, and the client establishes the connection with Telserv.

This section contains the following information:

- How to run a TELNET client in the OSS environment
- How to run a TELNET client in the NonStop Kernel environment

See Starting the Telserv Process on page 3-1 for information on creating a Telserv process.

Running a TELNET Client in the OSS Environment

Access to the Open System Services (OSS) environment, which is the HP NonStop POSIX offering, is available through TELNET clients running in NVT mode.

Enter the following command:

```
->TELNET hp-host-IP-address
```

After the connection has been made, the Welcome Banner and the Selection Menu are shown on the screen.

At the Enter Choice prompt, enter:

```
->TACL
```

At the TACL prompt, enter:

```
->LOGON user-id
```

You are prompted for your password. Enter your password.

To start a shell, enter:

```
->OSH
```

To exit from OSS, type the exit command followed by a zero (exit 0). This returns you to the TACL prompt. You can then log off to end the TELNET session

Note. Guardian applications can gain access to the OSS environment through OSSTTY as well as through TELNET. For information about OSSTTY, see the *Open System Services Management and Operations Guide* and the *Open System Services Programmer's Guide*. Information on OSSTTY is also contained in the following OSS manuals: *Open System Services System Calls Reference Manual, Open System Services Library Calls Reference Manual*, and *Open System Services Shell and Utilities Reference Manual*.

The remainder of this section discusses running the TELNET client in the NonStop™ Kernel environment.

Running a TELNET Client in the NonStop™ Kernel Environment

To run the TELNET client, enter the TELNET command. The form of the TELNET command is:

```
telnet [ / run-option [ , run-option ] ... / ]
       [ [ -s IP-address ] host [ port ] ]
```

run-option

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

-s *IP-address*

specifies one of the subnet IP addresses that the user has configured for the TCP/IP process and wants to use to connect to a remote host. If the user specifies an invalid IP address, TELNET displays an error message and stops.

host

identifies the remote host system. You can specify *host* as a host name or host IP address. See “Host Names” and Addressing Remote Hosts on page 8-5 for information on how to obtain host names and how to address remote hosts.

If you do not specify *host*, the TELNET client places you in command mode and displays the TELNET prompt:

```
telnet>
```

You can enter TELNET client commands at this prompt. TELNET client commands are described in TELNET Client Commands on page 9-1.

If you specify a *host*, the TELNET client 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 appear.

port

identifies the TCP port number. The default (well-known) port number is 23. You must specify the port number of the Telserv process you wish to communicate with.

For example, if the Telserv process you wish to connect to is running in port 6000 in a host named homesys, you would enter the following command:

```
TACL 4> telnet homesys 6000
WELCOME TO homesys.Tandem.COM [WINDOW $ZTNT.#xxxnnnn]
T9553D30 TELNET SERVER  01OCT94


Available Services:

TACL      EXIT
Enter Choice>
```

If you request a connection to a port that does not have a running Telserv process, you will not get a TELNET connection.

Establishing a Connection

When you request a connection to Telserv, the following steps show an example of the tasks Telserv performs to establish a connection with the client:

1. Creates a dynamic window for the client; for example, $ZTNT.#PTY05.
2. Negotiates TELNET options with the client.
3. Sends to the client a Welcome Banner that indicates the system name on which the server is running and the terminal name.
4. Displays on your terminal the Service Menu which lists the 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.
5. If you choose a TACL, 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 [port $ztc0 #23 window..$ztn0#pty0000]
Telserv  - T9553d30  - (31OCT94)
Available Services:
TACL  EXIT
Enter Choice> tacl
       .
       .
TACL 1>
```

Once a connection is established, you can use the TELNET client commands. TELNET Client Commands and TELNET Commands on page 9-1 describes these commands. Using the TELNET client toggle command, you can choose the escape option and issue TELNET commands. TELNET Commands on page 9-9 describes the TELNET commands.

Host Names and Name-Resolution Files

A host name is the official name by which the host system is known to the Internet. On a NonStop S-series system, 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, the mode (INET, INET6 or DUAL) in which you are running.

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.

The TACL ADD DEFINE command that follows is an example:

```
2> ADD DEFINE =TCPIP^HOST^FILE, FILE $SYSTEM.ZTCPIP.HOSTS
```

You must issue such an ADD DEFINE command to indicate that a HOSTS file is to be used, as well as the name of the desired HOSTS file. Otherwise, the DNR assumes it must use DNS and consults the RESCONF file.

Note that you also must set the TCPIP^HOST^FILE parameter at each terminal that uses the TCP/IP network. Then, when you invoke a TCP/IP application with reference to a host name, the DNR uses the appropriate HOSTS file. For convenience, include such an ADD DEFINE command as an entry in the TACLCSTM file, so that the command executes automatically every time you log onto the NonStop system.

For more information about the HOSTS file, see these 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. H-series and J-series systems do not support 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 Telserv in conjunction with NonStop TCP/IPv6 or CIP, 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 example,

```
2> ADD DEFINE =TCPIP^NODE^FILE, FILE $SYSTEM.ZTCPIP.IPNODES
```

There is no sample IPNODES file on the SUT; you must create this file yourself if you want to use it. The format of an IPNODES file is the same as that of a HOSTS file.

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

Addressing Remote Hosts

You can address a remote host by specifying either a host Internet address or a host name. When you start Telserv with NonStop TCP/IPv6 or CIP serving as the underlying transport protocol, you can use IPv6 addresses as well as IPv4 addresses. Ask the system administrator for the Internet address or name of the host system you want to use.

Host Internet Address (IPv4)

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

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

For example, the class A address `79.3.8.22` identifies the network address as `79` and the local host address as `3.8.22`.

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

Sometimes an Internet 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 on host Internet addresses, see the *TCP/IP Configuration and Management Manual.*

Host Internet Address (IPv6)

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, you do not need to enter those zeros. 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, you can use double colons (::) 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*.

9 TELNET Client Commands and TELNET Commands

TELNET Client Commands

This subsection contains descriptions of the syntax (enclosed in boxes) and rules for using TELNET client commands and provide examples of typical ways to use the commands.

Table 9-1 summarizes the TELNET client commands.

Table 9-1. TELNET Client Command Summary

Command	**Purpose**
close	Disconnects from remote system and returns to TELNET prompt
display	Displays Telserv special characters and current settings of toggle controls
help *or* ?	Displays information on Telserv commands
mode	Sets the input mode
open	Connects to the remote system
quit	Disconnects from the remote system and exits TELNET
send	Sends special characters to the remote system
status	Displays current status of TELNET
toggle	Turns TELNET toggle controls on or off
ttywritesz	Specifies number of characters to be sent at one time from the remote system to your terminal

To enter one of these commands, you need only 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, 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 that indicates the available services. If the system indicates that the service you select is not available, ask the system manager of the remote system to start the service.

close Command

Use the close command to disconnect from the remote system and return to the TELNET prompt at the local system.

```
close
```

Examples

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 Command

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. You need only specify enough characters to uniquely identify the argument as shown by the letters in parentheses following each 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.

argument can be any one of the following terms.

Argument	**Display**
autoflush (autof)	will flush output when sending interrupt characters
autosynch (autos)	won't send interrupt characters in urgent mode
crmod (c)	won't map carriage return on output
echo (ec)	[^E] echo
eof (eo)	[^Y] eof
erase (er)	[^H] erase
escape (es)	[^]] escape

Argument	Display
`flushoutput (f)`	[^O] flushoutput
`interrupt (i)`	[^C] interrupt
`kill (k)`	[^X] kill
`localchars (l)`	won't recognize certain control characters
`netdata (n)`	won't print hexadecimal representation of network traffic
`options (o)`	won't show option processing
`quit (g)`	[^\] quit

Example

To display the interrupt character and the current setting of interrupt and netdata, enter the following display command:

```
telnet> display i n
[^C]  interrupt.
Won't print hexadecimal representation of network traffic.
```

help or ? Command

Use the help or ? command to display either a summary of all TELNET commands or information on a specific command.

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

command-name

specifies the name of the command you want described.

Examples

To request information on the open command, enter the following command:

```
telnet> help open
connect to a site
```

To request information on the toggle command, enter the following command:

```
telnet> ? toggle
toggle operating parameters ('toggle ?' for more)
```

mode Command

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, the TELNET client 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.

The TELNET client 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 Command

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. See Addressing Remote Hosts on page 8-5 for information on specifying host names and addresses.

port

specifies the number of a port on the remote system that you want the TELNET client to contact. If you omit *port*, the TELNET client attempts to contact a TELNET server at the default (well-known) port, which is 23.

Example

The following command establishes a connection to the TELNET server on a system named dist101:

```
telnet> open dist101
```

quit Command

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 Command

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 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. The arguments and their corresponding TELNET sequences are:

Argument	TELNET Sequence Sent
`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`	Are You There sequence.
`brk`	Break sequence.
`ec`	Erase Character sequence, which asks the remote system to erase the last character you entered.
`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`	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`	Interrupt process sequence, which asks the remote system to abort the currently running process.
`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 Command

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 Command

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. The following table describes each toggle and its initial setting when you start TELNET.

Argument	**Function**
autoflush	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, the TELNET client 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. The initial setting is ON.
autosynch	Determines whether the TELNET SYNCH sequence is sent. If both autosynch and localchars are on when you enter an intr character, the TELNET client sends the SYNCH sequence and previously typed data is flushed. The initial setting is OFF.
crmod	Controls carriage return mode (crmod). When crmod is on, carriage return 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. The initial setting is OFF.
localchars	Controls the local recognition of special characters and their transformation into appropriate TELNET sequences. 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. The initial setting is OFF.
netdata	When netdata is on, all network data is displayed in hexadecimal format. This toggle is useful for debugging. The initial setting is OFF.
options	When options is on, internal TELNET protocol processing that relates to TELNET options is displayed. This toggle is useful for debugging. The initial setting is OFF.

The special characters used by the localchars toggle are:

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, sends the TELNET EC (erase character) sequence to the remote system.
escape	^]	Sends the TELNET escape character. You use these keys to call up the TELNET prompt for TELNET command mode. See TELNET Commands on page 9-9.
flushoutput	^O	Sends the TELNET AO (abort output) sequence to the remote system.

Name	**Keys**	**Function**
`interrupt`	^C	Sends the TELNET IP (interrupt processing) sequence to the remote system.
`kill`	^X	In character-input mode, sends the TELNET EL (erase line) sequence to the remote system.
`quit`	^\	Sends the TELNET BRK (break) sequence to the remote system.

* The ^ represents the CNTL command.

`?`

displays information on the toggle command.

Examples

To change the setting of the autoflush and localchars toggles, enter the following command:

```
telnet> toggle autof l
```

To display information on 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>
```

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:
```

ttywritesz Command

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
```

TELNET Commands

Table 9-2 describes the TELNET commands.

Table 9-2. TELNET Commands

Command Code	Function
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.
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 suboption. The terminal type is negotiated under SB.
DO	Signifies that either the client or the server is asking the other to use a certain option.

Table 9-2. TELNET Commands

Command Code	Function
DONT	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.
WILL	Signifies that either the client or the server is willing to use a certain option.
WONT	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 9-3:

Table 9-3. Negotiable TELNET Options Using DO, DONT, WILL, and WONT

Option	Result
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.
LM	Line Mode. Use local editing and send complete line instead of individual 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.)
TM	Time Mark. Verifies that transmitted data has been completely processed.
TT	Terminal Type. Exchanges information on the make and model of the terminal being used.

Part IV. Programming Information

10 Modes of Operation

Several modes of communication are used between:

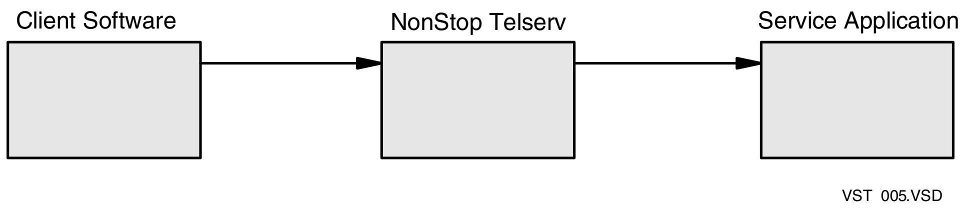
- An application on the NonStop system and the Telserv process.

Note. The term application refers to any service application that you write and that the interactive user can select, as well as to such HP NonStop utilities such as TACL, FUP, and INSPECT.

- The Telserv process and the client software, including any TELNET client software you write.

Figure 10-1 shows the relationship among a client, the Telserv process, and a service application

Figure 10-1. Client Software, the Telserv Process, and a Service Application



Conversation Mode and Block Mode

The HP NonStop Kernel defines two modes of operation: conversation mode and block mode. Conversation mode configures a conversational terminal service and initiates the session in conversation mode. This is the default service type. Telserv allows the use of a type-ahead feature while in conversation mode. Telserv echoes each character, even if the application has not posted a read request. Because of the type-ahead feature, Telserv might echo the service name or window name sent by the user or client program even before the Telserv process sends its own banner. (Note that type ahead is not provided by Termprocess/ATP6100.)

Block mode configures a 6530 terminal-type block-mode service and initiates the session in block mode. The application may change the mode by using the file-system SETMODE procedure. Client software is included in terminal emulators from other vendors, such as Sun Microsystems. This software can run in block mode or conversation mode. You can also write your own client program that runs in both modes. The TELNET client supplied by HP runs in NVT mode and does not support 6530 terminal types.

For information about SETMODE procedures, see the *Guardian Procedure Calls Reference Manual*.

If Telserv sets the 653X emulation mode, which includes block mode, the capability to switch back and forth between block mode and conversation mode is available. The client software uses option TELOPT_TTYTYPE during negotiation and sends one of the following strings:

```
TN6530
TN653X
UB6530
UB653X
```

In block mode, additional applications are available to the terminal user; for example, TEDIT on a NonStop S-series server operates in block mode.

Telserv ignores all other terminal-type options requested during negotiation with a client program.

Line Mode and Character Mode

In character mode, the TELNET client immediately sends each character the user types to the remote system. In line mode, the text the user types is echoed locally and sent to the remote host when the terminal user presses the RETURN key (start and stop bits indicate the beginning and end of the data string). The TELNET client enters the mode the user specifies only if the remote host is capable of entering that mode. By default, the mode is CHARACTER.

ASCII Mode and Binary Mode

Telserv supports both ASCII and Binary transfer modes. ASCII is the default mode. When in ASCII mode, Telserv performs end-of-line processing of incoming and outgoing data.

Telserv can provide binary data transfer when the client software and the application on the NonStop system perform the actions described below.

Client Software

The client must initiate TELNET Binary mode for both directions. This action results in turning off end-of-line processing. However, incoming and outgoing <CR><NULL> and <CR><LF> remain unchanged.

Note. For the negotiation to succeed, the Telserv process must be in 8-bit mode. By default, Telserv runs in 8-bit mode.

To set 8-bit mode, the system administrator who starts the process can use the -8 start-up option, or the application can issue a SETMODE 23 procedure call.

Failure of the SETMODE 23 requires system administrator intervention. The system administrator must restart Telserv and either explicitly specify -8 for the start-up option or allow the option to default to 8-bit character mode.

Even when Binary mode is set, both the Telserv process and the client software must escape IAC so as not to conflict with TELNET commands

Application

The application on the NonStop system must issue the following SETMODE procedure calls:

SETMODE 8,0,0	Sets conversation mode, which is the only mode allowed for NVT.
SETMODE 6,0,0	Specifies no spacing, thus preventing <CR><LF> from being added to the output stream.
SETMODE 20,0,0	Specifies no echoing, thus preventing characters received from being echoed.
SETMODE 9 *nnnn*, *nnnn*	Sets the interrupt characters. The values for *nnnn* vary and are established by the programmer of the application. See the *Guardian Procedure Calls Reference Manua*l.
SETMODE 23,3,0	Sets character size to 8 bits. This SETMODE is equivalent to having the system administrator start Telserv with the -8 option.

A

Telserv Error Messages Sent to Terminals

Recovering From Errors

You can apply the following general approach to recover from errors that you receive on your terminal screen while using TELNET with the Telserv Product.

- Make sure that 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 administrator 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. To find a complete explanation of these errors, refer to the *TCP/IP Programming Manual*. The descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual* also include some *error-reason* values you might encounter, as well as the *error-number* values listed in some messages.
- In some cases, errors occur because the TCP/IP process is no longer running. Ask your system administrator to check whether the TCP/IP process has stopped.

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 that you have specified the host name or address correctly. If the TELNET port is not 23, specify the correct port number.

```
Can't set mode number on term err error-number
```

Cause. TELNET cannot set the mode you specified because of the error specified by *error-number*.

Effect. The mode is not set.

Recovery. See Recovering From Errors at the beginning of this appendix.

```
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" at the beginning of this appendix.

```
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 at the beginning of this appendix.

```
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 at the beginning of this appendix.

```
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 to reconnect again later.

```
Open of my-term error error-number
```

Cause. A file-system error occurred when TELNET tried to open the virtual terminal file process.

Effect. The connection that you requested is not made.

Recovery. See Recovering From Errors at the beginning of this appendix.

```
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 failed.

Recovery. See “Recovering From Errors” at the beginning of this appendix.

```
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 administrator if the TCP/IP process is running. See “Recovering From Errors” at the beginning of this appendix.

```
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 administrator if the TCP/IP process is running. See Recovering From Errors at the beginning of this appendix.

```
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.

Ask your system administrator if the TCP/IP process is running. See “

```
TELNET: bind-nw failed wiht error no.: error-code
```

Cause. TELNET could not bind to the specified subnet IP address because of the following error: *error-code*.

Effect. The run command is not executed.

Recovery. Retry the command. Make sure that you are using the correct subnet IP address.

```
TELNET: connect error-reason
```

Cause. TELNET was unable to connect to the host for the reason specified by *error-reason*.

Effect. The connection that you requested is not established.

Recovery. See Recovering From Errors at the beginning of this appendix.

```
TELNET: getaddrinfo: error-message
```

Cause. TELNET detected an internal error while resolving the host name and service that the user entered.

Effect. TELNET terminates.

Recovery. Contact your system administrator.

```
TELNET 'send' error - argument disappeared!
```

Cause. TELNET detected an internal error while parsing the send command that you entered.

Effect. The command is not executed.

Recovery. Try entering the command again. Make sure that you are using the correct syntax.

```
TELNET: tcp/telnet unknown service
```

Cause. The entry for TELNET is missing from the $SYSTEM.ZTCPIP.SERVICES file, so TELNET cannot determine which terminal control process (TCP) port to use.

Effect. The command failed.

Recovery. Ask the system administrator 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 that 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 that 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 that you specified is invalid.

Effect. The command is ignored.

Recovery. Enter the command again and specify “c” (for character) or “l” (for line).

```
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 failed.

Recovery. See Recovering From Errors at the beginning of this appendix.

Telserv Error Messages

```
Sorry, too many users
```

Cause. No windows are available. The number of virtual terminals, or windows, has exceeded the limit set for the process by the SCF MAXTERMINALS attribute, and only dynamic services are configured.

Effect. Service selection is not performed and the connection terminates.

Recovery. As a terminal user, you can take no direct action. Ask the system administrator or operator to either increase the MAXTERMINALS limit for the process or free some windows.

```
Sorry, Login session aborted... Try later
```

Cause. Either the Login object is not licensed, or Login could not launch the application. Failure to launch could be caused by several problems. For example, Telserv might be busy, or the application object might be corrupted.

Effect. You cannot launch the application.

Recovery. As a terminal user, you can take no direct action. Ask the system administrator to:

- License the Login object if it is not licensed (see the LICENSE command in the *File Utility Program (FUP) Reference Manual*), or
- Replace the application object, or
- Check to see whether related CPUs are down.

```
***WARNING*** Terminal will be disconnected if it stays idle
```

Cause. There has been no activity at this terminal for too long.

Effect. The user is disconnected after repeated warning messages.

Recovery. Maintain activity on the terminal, or ask the system administrator to use SCF to disable the banner time-out value or time-out value for the process.

```
Terminal was idle too long! Disconnecting...
```

Cause. There has been no activity at this terminal for a long period.

Effect. The connection to the system is terminated. This error message is sent after several warning messages.

Recovery. Connect again and maintain activity on the terminal, or ask the system administrator to use SCF to disable the banner time-out value or the time-out value for the process.

```
Sorry, none of the CPUs configured are available.
Trying to launch on available CPUs...
```

Cause. All CPUs specified in the CPULIST attribute of the SCF PROCESS object are down.

Effect. The application is launched on another available CPU.

Recovery. As a terminal user, you can take no direct action. Ask the operator to bring up the CPU s specified in the CPULIST attribute or ask the system administrator to modify CPULIST so that it specifies available CPUs.

```
Sorry, cannot launch login server
```

Cause. The Telserv process could not launch the login server because of a memory allocation failure or because Telserv is busy.

Effect. You cannot launch the chosen application.

Recovery. As a terminal user, you can take no direct action. Ask the system administrator to reduce the load on Telserv or free up some memory.

```
Aborting window due to internal server error..
```

Cause. Internal data corruption occurred in the Telserv process.

Effect. Neither the Welcome Banner nor the Service Menu appears.

Recovery. Try connecting again. If the problem persists, ask the operator to restart the Telserv process. Contact the system administrator.

```
Sorry, window cannot be selected
Sorry, invalid choice
```

Cause. You typed in a window name instead of a service name.

Effect. The Service Menu is displayed again.

Recovery. Enter the correct service name rather than a window name. Note that window names begin with a pound sign (#).

```
Sorry, too many users for service: <service-name>
```

Cause. The number of windows allocated to this service, have been exhausted. (The service is of subtype STATIC.)

Effect. You cannot select this service. The Service Menu is displayed again

Recovery. As a terminal user, you can take no direct action. Ask the system administrator to add more windows for this service or free some windows that currently have this service.

```
Sorry, service <service-name> has no started subordinated
windows
```

Cause. None of the windows related to the service are in the STARTED state, but windows do exist for this service.

Effect. You cannot launch the chosen application.

Recovery. As a terminal user, you can take no direct action. Ask the system administrator to start the windows related to the service.

```
Sorry, too many users for default service: <service-name>
```

Cause. The number of windows allocated to this service, which is the default service, has been exhausted.

Effect. The Service menu is displayed.

Recovery. As a terminal user, you can take no direct action. Ask the system administrator either to:

- Use SCF to add more windows for this service, or
- Free some windows which currently have this service

```
Sorry, too many users for dynamic service: <service-name>
```

Cause. The number of windows has exceeded the limit set for the process by the SCF MAXTERMINALS attribute, and the service chosen has the SUBTYPE attribute specified as DYNAMIC.

Effect. The terminal is disconnected.

Recovery. Try again after asking the system administrator either to:

- Increase the MAXTERMINALS limit for the process, or
- Free some windows

B SCF Error Messages for Telserv

This appendix provides you with information for interpreting Telserv subsystem error messages generated by the Telserv process. Italicized words in the message descriptions stand for values that are inserted by the Telserv process.

Error messages are identified by a 5-digit number preceded by the character E. You can use the SCF HELP command to display the text of a message; for example, when the system displays:

```
TELSERV E00001
```

You can enter the following command to display the text of the message:

```
-> HELP TELSERV 1
```

For detailed information on error messages, see the appendix on error numbers and error lists in the *TCP/IP TELNET Management Programming Manual*.

1

```
TELSERV E00001 Invalid attribute value: name for object-name
```

Cause. A command was issued with an illegal attribute value.

Effect. The command does not execute.

Recovery. Issue the command again with values within the valid range.

2

```
TELSERV E00002 object-name cannot be stopped because it has
               active opens.
```

Cause. The STOP command was issued against an object with active opens.

Effect. The command does not execute.

Recovery. Either stop all processes that have the specified window open and issue the STOP command again, or issue the ABORT command. The ABORT command stops windows without regard to any process that has the window open. This response might cause the processes with open windows to fail.

3

```
TELSERV E00003 Inconsistency between SERVICETYPE and
               SERVICENAME.
```

Cause. SERVICETYPE in this ADD command is different from the SERVICETYPE associated with the SERVICENAME already defined.

Effect. The command does not execute.

Recovery. Change the SERVICETYPE or SERVICENAME in this ADD command to be consistent with each other.

4

```
TELSERV E00004 Inconsistency between attribute values: name1
               and name2
```

Cause. Specified attribute values are within range, but are not valid in the presence of other attributes. For example, CPU is not a valid attribute unless PROGRAM is specified.

Effect. The command does not execute.

Recovery. Make sure that the CPU, PRI, PARAM, SWAP and LIB attributes are used with PROGRAM, and the ACCESS attribute is used with OWNER.

5

```
TELSERV E00005 Subordinate objects active
```

Cause. Subordinate objects are in the started state.

Effect. The command does not execute.

Recovery. Stop or abort subordinate objects.

6

```
TELSERV E00006 Cannot add window to a dynamic service.
```

Cause. The SERVICENAME specified in the ADD WINDOW command is a dynamic service. A dynamic service automatically creates a window when a session is established.

Effect. The command does not execute.

Recovery. Make sure that the SERVICENAME specified in the ADD WINDOW command is a static service.

7

```
TELSERV E00007 Special Service name may not be altered or
               deleted.
```

Cause. Special Services have names that start with the character “Z”. These services cannot be altered or deleted.

Effect. The command does not execute.

Recovery. Informative message only; no corrective action is needed. The Special Service remains unchanged.

8

```
TELSERV E00008 Subordinate object exists for name.
```

Cause. Service cannot be deleted, or altered from STATIC to DYNAMIC, when there are subordinate windows configured.

Effect. The command does not execute.

Recovery. ABORT and DELETE subordinate windows first.

9

```
TELSERV E00009 No opener found for name.
```

Cause. There are no openers for the object specified in the LISTOPENS command.

Effect. The command does not execute.

Recovery. Informative message only; no corrective action is needed.

10

```
TELSERV E00010 Invalid file or device name specified for
               attribute attribute.
```

Cause. The format of the file or device name is incorrect.

Effect. The command does not execute.

Recovery. Reenter the command and specify the correct file or device name.

11

```
TELSERV E00011 File or device name specified for attribute
               attribute does not exist.
```

Cause. The specified file or device name does not exist.

Effect. The command does not execute.

Recovery. Reenter the command and specify the correct file or device name.

12

```
TELSERV E00012 File specified for attribute attribute is not
               an executable program.
```

Cause. The file code of the specified file indicates that it is not an executable program. Executable program files have file code 100. The wrong file was probably specified.

Effect. The command does not execute.

Recovery. Reenter the command and specify the correct file name.

13

```
TELSERV E00013 Limit exceeded on number of configured windows
               for resilient service.
```

Cause. Only one configured window is allowed for a resilient service. An attempt to add more that one window to a resilient service, or attempt to alter service to resilient for a service which has more than one window configured has been rejected.

Effect. The command does not execute.

Recovery. For the ADD WINDOW, add a new resilient service and add the window under the new service. For the ALTER SERVICE command, delete the configured windows until only one window is left, and then retry the ALTER SERVICE command.

14

```
TELSERV E00014 Response buffer too small.
```

Cause. The read count in the WRITREAD request is less than the recommended response buffer length.

Effect. The command does not execute.

Recovery. Specify a read count greater than or equal to ZCOM-VAL-BUFLEN. Make sure the buffer you are using is large enough.

15

```
TELSERV E00015 Specified caller IP address already exits
               in another service.
```

Cause. The specified caller IP address is already associated with an existing window in a different service.

Effect. The command does not execute.

Recovery. Specify a different caller IP address.

16

```
TELSERV E00016 Caller IP address is required for window.
```

Cause. Caller IP address must be specified for windows subordinate to a service whose ASSIGNED attribute has been turned on.

Effect. The command does not execute.

Recovery. Specify caller IP address for the window.

17

```
TELSERV E00017 This assigned attribute is not supported
               in the IPX/SPX environment.
```

Cause. An invalid command was entered in the IPX/SPX environment.

Effect. The command does not execute.

Recovery. Use TCP/IP as the TELSERV transport protocol.

18

```
TELSERV E00018 The maximum number of services has been
               exceeded.
```

Cause. The current maximum number of services is 256.

Effect. The command does not execute.

Recovery. To add a new service delete an existing service or wait till the service count is below 256.

19

```
TELSERV E00019 The Default Service already exists.
```

Cause. Only one Default Service can be allowed per Telserv process.

Effect. The command does not execute.

Recovery. Use another Telserv process to configure the Default Service.

20

```
TELSERV E000020 Attribute already specified.
```

Cause. An attribute is being specified more than once.

Effect. The command does not execute.

Recovery. Reissue the command after removing the duplicate attribute.

21

```
TELSERV E000021 Missing required attribute.
```

Cause. A required attribute for the command has not been specified.

Effect. The command does not execute.

Recovery. Re-issue the command with the missing required attribute. See Recovery in error message 4 for an explanation of attributes that must accompany others.

22

```
TELSERV E000022. PM Internal error -’Case’ statement out of
range.
```

Cause. Internal data structure error.

Effect. The command does not execute.

Recovery. Collect a log of the commands issued during the session and contact the HP support center.

C SCF Command Syntax

This appendix provides a summary of the SCF command syntax.

ABORT Command

```
ABORT [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

ABORT PROCESS

```
ABORT [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

ABORT WINDOW

```
ABORT [ / OUT file-spec / ] WINDOW window-name
```

ADD Command

```
ADD [ / OUT file-spec / ] [ object-spec ]

   { , attribute-spec [ , attribute-spec ] ... }
```

ADD SERVICE

```
ADD [ / OUT file-spec / ] SERVICE service-name

   { , attribute-spec [ , attribute-spec ] ... }
```

where *attribute-spec* is:

```
[ ACCESS { ALL | SYSTEM | OWNER | NONE } ]
[ ASSIGNED { ON | OFF }                  ]
[ AUTODELETE { ON | OFF }                ]
[ CPU { 0 through 15 }                   ]
[ DEFAULT { ON | OFF }                   ]
[ DISPLAY { ON | OFF }                   ]
[ LIB file-name                          ]
[ OWNER "24-char"                        ]
[ PARAM "128-char"                       ]
[ PRI { 0 through 190 }                  ]
[ PROGRAM file-name                      ]
[ RESILIENT { ON | OFF }                 ]
[ SUBTYPE { DYNAMIC | STATIC }           ]
[ SWAP file-name                         ]
{ TYPE { BLOCK | CONVERSATION | PRINT }  }
```

ADD WINDOW

```
ADD [ / OUT file-spec / ] WINDOW window-name

   { , attribute-spec [ , attribute-spec ] ... }

where attribute-spec is:

[ CALLER "IP4-address"                             ]
[ CALLER6 "IP6-address"                            ]
[ ENDOFFILE 0 through 255                          ]
[ ENDOFLINE 0 through 255                          ]
[ ERASE 0 through 255                              ]
[ INTERRUPT 0 through 255                          ]
[ LINEKILL 0 through 255                           ]
[ SERVICETYPE { BLOCK | CONVERSATION | PRINT } ]
{ SERVICENAME "service-name" }
```

ALTER Command

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

   { , attribute-spec [ , attribute-spec ] ... }
```

ALTER PROCESS

```
ALTER [ / OUT file-spec / ] PROCESS process-name

   { , attribute-spec [ , attribute-spec ] ... }

where attribute-spec is:

[ MAXTERMINALS { 1 through 256 }          ]
[ MENU { ON | OFF }                       ]
[ TIMEOUTVALUE { 3 through 32767 }        ]
[ BANNERTIMEOUTVALUE { 3 through 32767 } ]
[ CPULIST {0,1,2...,15 } ]
```

ALTER SERVICE

```
ALTER [ / OUT file-spec / ] SERVICE service-name

   { , attribute-spec [ , attribute-spec ] ... }

where attribute-spec is:

[ ACCESS { ALL | SYSTEM | OWNER | NONE } ]
[ ASSIGNED { ON | OFF }                  ]
[ AUTODELETE { ON | OFF }                ]
[ CPU { 0 through 15 }                   ]
[ DEFAULT { ON | OFF }                   ]
[ DISPLAY { ON | OFF }                   ]
[ LIB file-name                          ]
[ OWNER "24-char"                        ]
[ PARAM "128-char"                       ]
[ PRI { 0 through 190 }                  ]
[ PROGRAM file-name                      ]
[ RESILIENT { ON | OFF }                 ]
[ SUBTYPE { DYNAMIC | STATIC }           ]
[ SWAP file-name                         ]
[ TYPE { BLOCK | CONVERSATION | PRINT }  ]
```

ALTER WINDOW

```
ALTER [ / OUT file-spec / ] WINDOW window-name

   { , attribute-spec [ , attribute-spec ] ... }

where attribute-spec is:

[ CALLER "IP4-address"     ]
[ CALLER6 "IP6-address"    ]
[ ENDOFFILE 0 through 255 ]
[ ENDOFLINE 0 through 255 ]
[ ERASE 0 through 255     ]
[ INTERRUPT 0 through 255 ]
[ LINEKILL 0 through 255  ]
```

DELETE Command

```
DELETE [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

DELETE SERVICE

```
DELETE [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

DELETE WINDOW

```
DELETE [ / OUT file-spec / ] WINDOW window-name
```

INFO Command

```
INFO [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

INFO PROCESS

```
INFO [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

INFO SERVICE

```
INFO [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

INFO WINDOW

```
INFO [ / OUT file-spec / ] WINDOW window-name [ , DETAIL ]
```

LISTOPENS Command

```
LISTOPENS [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

LISTOPENS PROCESS

```
LISTOPENS [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

LISTOPENS WINDOW

```
LISTOPENS [ / OUT file-spec / ] WINDOW window-name
```

NAMES Command

```
NAMES [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ]
```

NAMES PROCESS

```
NAMES [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

NAMES SERVICE

```
NAMES [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ]
```

NAMES WINDOW

```
NAMES [ / OUT file-spec / ] WINDOW window-name
```

NAMES *null*

```
NAMES [ / OUT file-spec / ] process-name
```

START WINDOW Command

```
START [ / OUT file-spec / ] WINDOW window-name
```

STATS Command

```
STATS [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ] [ , RESET ]
```

STATS PROCESS

```
STATS [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , RESET ]
```

STATS SERVICE

```
STATS [ / OUT file-spec / ] SERVICE service-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , RESET ]
```

STATS WINDOW

```
STATS [ / OUT file-spec / ] WINDOW window-name [ , RESET ]
```

STATUS Command

```
STATUS [ / OUT file-spec / ] [ object-spec ]

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

STATUS PROCESS

```
STATUS [ / OUT file-spec / ] PROCESS process-name

   [ , SUB [ NONE | ALL | ONLY ] ] [ , DETAIL ]
```

STATUS WINDOW

```
STATUS [ / OUT file-spec / ] WINDOW window-name [ , DETAIL ]
```

STOP Command

```
STOP [ / OUT file-spec / ] [ object-spec ]
```

STOP PROCESS

```
STOP [ / OUT file-spec / ] PROCESS [ process-name ]
```

STOP WINDOW

```
STOP [ / OUT file-spec / ] WINDOW [ window-name ]
```

TRACE Command

```
TRACE [ / OUT file-spec / ] [ object-spec ]

   { , STOP                    }
   { [ , COUNT count         ]
     [ , NOCOLL              ]
     [ , PAGES pages         ]
     [ , RECSIZE size        ]
     [ , SELECT select-spec  ]
     [ , TO file-spec        ]              ***
     [ , WRAP                ] }
```

*** This attribute is required when a trace is started.

TRACE PROCESS

```
TRACE [ / OUT file-spec / ] [ object-spec ]

   { , STOP                   }
   { [ , COUNT count        ]
     [ , NOCOLL             ]
     [ , PAGES pages        ]
     [ , RECSIZE size       ]
     [ , SELECT select-spec ]
     [ , TO file-spec       ]             ***
     [ , WRAP               ] }
```

*** This attribute is required when a trace is started.

where *select-spec* is one or more of the following specifications:

```
{ keyword                      }
{ ( keyword [ , keyword ] ...) }

{ number                     }
{ ( number [ , number ] ...) }
```

where *keyword* or *number* is one or more of the following values:

```
[ ALL    ] [ -1 ]
[ CONN   ] [  2 ]
[ LOGIN  ] [  9 ]
[ ONLINE ] [ 14 ]
[ OPEN   ] [  4 ]
[ RECV   ] [ 10 ]
[ SCB    ] [  8 ]
[ SPI    ] [ 13 ]
[ TELOPT ] [ 15 ]
```

TRACE WINDOW

```
TRACE [ / OUT file-spec / ] [ object-spec ]

   { , STOP                     }
   { [ , COUNT count          ]
     [ , NOCOLL               ]
     [ , PAGES pages          ]
     [ , RECSIZE size         ]
     [ , SELECT select-spec ]
     [ , TO file-spec         ]              ***
     [ , WRAP                 ] }
```

*** This attribute is required when a trace is started.

where *select-spec* is one or more of the following specifications:

```
{ keyword                       }
{ ( keyword [ , keyword ] ...) }

{ number                      }
{ ( number [ , number ] ...) }
```

where *keyword* or *number* is one or more of the following values:

```
[ ALL    ] [ -1 ]
[ ONLINE ] [ 22 ]
[ RCB    ] [ 21 ]
[ SOCK   ] [ 18 ]
[ TTY    ] [ 20 ]
```

VERSION Command

```
VERSION [ / OUT file-spec / ] [ object-spec ] [ , DETAIL ]
```

VERSION PROCESS

```
VERSION [ / OUT file-spec / ] PROCESS [ process-name ]

   [ , DETAIL ]
```

VERSION *null*

```
VERSION [ / OUT file-spec / ] [ process-name ] [ , DETAIL ]
```

Index