TCP/IP Configuration and Management Manual

Abstract

This manual describes how to set up and manage the HP NonStop™ Transmission Control Protocol/Internet Protocol (TCP/IP) subsystem on an HP NonStop S-series server and on an HP Integrity NonStop NS-series server. This manual is written for system managers, operators, and others who configure and manage NonStop TCP/IP.

Product Version

NonStop TCP/IP G06
NonStop TCP/IP H01

Supported Release Version Updates (RVUs)

This manual supports G06.25 and all subsequent G-series RVUs and H06.03 and all subsequent H-series RVUs until otherwise indicated by its replacement edition.

Part Number	Published
427132-004	March 2014

Document History

Part Number	Product Version	Published
137045	Tandem NonStop TCP/IP G05	February 1998
140771	Tandem NonStop TCP/IP G06	May 1998
427132-001	NonStop TCP/IP G06	August 2002
427132-002	NonStop TCP/IP G06	August 2004
427132-003	NonStop TCP/IP G06 NonStop TCP/IP H01	July 2005
427132-004	NonStop TCP/IP G06 NonStop TCP/IP H01	March 2014

New editions incorporate any updates issued since the previous edition.

Legal Notices

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Export of the information contained in this publication may require authorization from the U.S. Department of Commerce.

Microsoft, Windows, and Windows NT are U.S. registered trademarks of Microsoft Corporation.

Intel, Itanium, Pentium, and Celeron are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Motif, OSF/1, UNIX, X/Open, and the "X" device are registered trademarks and IT DialTone and The Open Group are trademarks of The Open Group in the U.S. and other countries.

Open Software Foundation, OSF, the OSF logo, OSF/1, OSF/Motif, and Motif are trademarks of the Open Software Foundation, Inc.

> OSF MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE OSF MATERIAL PROVIDED HEREIN, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
>
> OSF shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance, or use of this material.
>
> © 1990, 1991, 1992, 1993 Open Software Foundation, Inc. This documentation and the software to which it relates are derived in part from materials supplied by the following:
>
> © 1987, 1988, 1989 Carnegie-Mellon University. © 1989, 1990, 1991 Digital Equipment Corporation. © 1985, 1988, 1989, 1990 Encore Computer Corporation. © 1988 Free Software Foundation, Inc. © 1987, 1988, 1989, 1990, 1991 Hewlett-Packard Company. © 1985, 1987, 1988, 1989, 1990, 1991, 1992 International Business Machines Corporation. © 1988, 1989 Massachusetts Institute of Technology. © 1988, 1989, 1990 Mentat Inc. © 1988 Microsoft Corporation. © 1987, 1988, 1989, 1990, 1991, 1992 SecureWare, Inc. © 1990, 1991 Siemens Nixdorf Informationssysteme AG. © 1986, 1989, 1996, 1997 Sun Microsystems, Inc. © 1989, 1990, 1991 Transarc Corporation.
>
> This software and documentation are based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. OSF acknowledges the following individuals and institutions for their role in its development: Kenneth C.R.C. Arnold, Gregory S. Couch, Conrad C. Huang, Ed James, Symmetric Computer Systems, Robert Elz. © 1980, 1981, 1982, 1983, 1985, 1986, 1987, 1988, 1989 Regents of the University of California.
>
> Printed in the US

TCP/IP Configuration and Management Manual

Glossary | Index | Examples | Figures | Tables

3. Configuring the NonStop TCP/IP Subsystem

4. SCF Reference

Examples

Figures

Tables

What’s New in This Manual

Manual Information

Abstract

This manual describes how to set up and manage the HP NonStop™ Transmission Control Protocol/Internet Protocol (TCP/IP) subsystem on an HP NonStop S-series server and on an HP Integrity NonStop NS-series server. This manual is written for system managers, operators, and others who configure and manage NonStop TCP/IP.

Product Version

NonStop TCP/IP G06
NonStop TCP/IP H01

Supported Release Version Updates (RVUs)

This manual supports G06.25 and all subsequent G-series RVUs and H06.03 and all subsequent H-series RVUs until otherwise indicated by its replacement edition.

Part Number	Published
427132-004	March 2014

Document History

Part Number	Product Version	Published
137045	Tandem NonStop TCP/IP G05	February 1998
140771	Tandem NonStop TCP/IP G06	May 1998
427132-001	NonStop TCP/IP G06	August 2002
427132-002	NonStop TCP/IP G06	August 2004
427132-003	NonStop TCP/IP G06 NonStop TCP/IP H01	July 2005
427132-004	NonStop TCP/IP G06 NonStop TCP/IP H01	March 2014

New editions incorporate any updates issued since the previous edition.

New and Changed Information

This revision of the *TCP/IP Configuration and Management Manual* (427132-004) contains the following changes:

- Udpated TCP Layer Stats with new attributes in the section STATS PROCESS Display Format on page 4-67.

- Updated TCP Layer Stats with new attributes in the section Description of Statistics for the TCP Layer on page 4-69.

The earlier version of the *TCP/IP Configuration and Management Manual* contained the following changes:

- Manuals for that support the Integrity NonStop server have been added to About This Manual.
- Wherever SUBNET refers to the NonStop TCP/IP subsystem object, it is capitalized. Wherever subnet refers to the industry-standard networking term, it is not capitalized. See SUBNET Object Type on page 4-5.
- Information about Integrity NonStop servers has been added to:
    - Section 2, Overview of NonStop TCP/IP
    - Hardware and Software Requirements on page 2-8
    - Configuring an X25AM I/O Process on NonStop S-Series Systems and Integrity NonStop NS-series Systems on page 3-13
- Information about the new DNS product has been added to Starting the Domain Name Server (DNS) on page 3-30 and to Configuration Files for the Internet Environment on page 3-34.
- Figure 3-5, Domain Name System Hierarchy, on page 3-36 has been updated.

About This Manual

This manual describes G-series and H-series Release Version Updates (RVUs) of HP NonStop TCP/IP.

Who Should Use This Manual

This manual describes the installation, configuration, and management of the NonStop TCP/IP subsystem. It is for system managers, operators, and others who configure and manage the NonStop TCP/IP subsystem.

Required Background

This manual assumes familiarity with the standard TCP/IP family of protocols described in the RFCs and IENs and familiarity with configuring IP networks. Appendix B, NonStop TCP/IP Processes and Protocols, reviews these protocols but is not meant to replace reference books which present this information in depth. You should be familiar with NonStop system architecture, the networking product ServerNet LAN Systems Access (SLSA), Ethernet 4 ServerNet adapters (E4SAs), Token-Ring ServerNet adapters (TRSAs), Gigabit Ethernet ServerNet adapters (GESAs), Gigabit Ethernet 4-port ServerNet adapters (G4SAs), and Asynchronous Transfer Mode (ATM).

This manual also assumes that you are familiar with NonStop systems, including the NonStop operating system.

Prerequisite Materials

This subsection lists reference material that you can use to acquire the required background.

For an overview of TCP/IP, see the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

For an in-depth explanation of the Domain Name Server, see the book *DNS and BIND* by Paul Albitz and Cricket Liu, O’Reilly and Associates, Inc.

Request for Comments (RFC) is a series of documents published by the Internet Engineering Task Force (IETF). The following RFCs are referred to in that manual and can be located on the Internet at: http://ds.internic.net/ds/rfc-index.html.

- RFC 1577 “Classical IP and ARP Over ATM”
- RFC 768 “User Datagram Protocol”
- RFC 791 “Internet Protocol”
- RFC 792 “Internet Control Message Protocol”
- RFC 793 “Transmission Control Protocol”

- RFC 819 “Domain Naming Convention for Internet User Applications”
- RFC 821 “Simple Mail Transfer Protocol”
- RFC 826 “Ethernet Address Resolution Protocol”
- RFC 894 “Standard for the Transmission of IP Datagrams Over Ethernet Networks”
- RFC 973 (1034:1982, 1876, 1101; 1035: 1348, 1995, 1996
- RFC 974 “Mail Routing and Domain System”
- RFC 1034 (Now 1101, 1982 and 1876)
- RFC 1042 “Standard for the Transmission of IP Datagrams Over IEEE 802 Networks”
- RFC 1495 “Mapping Between X.400 and RFC-822 Message Bodies”

Background Manuals

- *Introduction to Networking for HP NonStop S-series Servers* provides an overview of NonStop server networking and data communications concepts, tasks, products, and manuals. It discusses ways to connect NonStop server subsystems to various devices and networks and it introduces the tools and interfaces you can use.
- *Introduction to Networking for HP Integrity NS-Series Servers* provides information about networking on the Integrity NonStop server, an overview of IP networking concepts, and networking and migration information.
- The *Guardian User’s Guide* provides basic information about the programs and utilities that are used most often in the Guardian environment by general system or application users and is aimed at beginning users of NonStop systems.

SLSA Adapters Manuals

For information about SLSA adapters, see:

Adapter Manuals
ATM Adapter Installation and Support Guide for the ATM3SA
Ethernet Adapter Installation and Support Guide for the E4SA
Fast Ethernet Adapter Installation and Support Guide for the FESA
Gigabit Ethernet Adapter Installation and Support Guide for the GESA
Gigabit Ethernet 4-Port Adapter Installation and Support Guide for the G4SA
Token-Ring Adapter Installation and Support Guide for the TRSA

NonStop S-Series Server Configuration Manuals

The following manuals provide the some of the required background for this manual:

- The *NonStop S-Series Planning and Configuration Guide* describes how to plan and configure a NonStop S-series server and provides a guide to other manuals for the NonStop S-series server.
- The *SCF Reference Manual for G-Series RVUs* describes the operation of SCF and the commands used to configure, control, and inquire about supported data communications subsystems.
- The *System Generation Manual for G-Series RVUs* describes how to use the SYSGENR program to create a new set of operating system files for a NonStop S-series server.
- The *LAN Configuration and Management Manual* describes the SLSA subsystem which provides parallel LAN I/O for NonStop S-series systems. In particular, read about logical interfaces (LIFs) and physical interfaces (PIFs) in that manual.

Integrity NonStop Server Configuration Manuals

The following manuals provide some of the required background for this manual:

- The *HP Integrity NonStop NS-Series Planning Guide* explains how to plan for new Integrity NonStop servers. In addition, this guide describes the ServerNet system area network (ServerNet SAN) and the available hardware and system configurations. It also provides a guide to other Integrity NonStop server manuals.
- The *SCF Reference Manual for H-Series RVUs* describes the operation of SCF and the commands used to configure, control, and inquire about supported data communications subsystems.
- The *LAN Configuration and Management Manual* describes the SLSA subsystem which provides parallel LAN I/O for NonStop S-series systems. In particular, read about logical interfaces (LIFs) and physical interfaces (PIFs) in that manual.

NonStop TCP/IP Core and Related Services Manuals

This manual refers to the following NonStop TCP/IP and other services manuals:

- The *TCP/IP Programming Manual* describes the programmatic interface to the NonStop TCP/IP data communications subsystem.
- The *TCP/IP Applications and Utilities User Guide* describes the interactive interfaces to the following NonStop TCP/IP applications: ECHO, FINGER, FTP, TFTP, TELNET, and TN6530. Server information is included for FTP, TFTP, and TELNET.
- The *Telserv Manual* describes the TELSERV SCF interface. This guide is intended for configuration and support planners who are responsible for the correct operation of the Telserv subsystem. This guide also provides information about the TN6530 terminal emulation utility.

- The *TCP/IP TELNET Management Programming Manual* describes the command/response interface and the EMS interface available to an application program for communication with the TCP/IP TELNET process.
- The *QIO Configuration and Management Manual* describes how to install and manage a QIO data communications subsystem. This manual also describes the SCF command used to configure, control, and inquire about the QIO subsystem
- The *Open System Services Shell and Utilities Reference Manual* documents the contents of the inetd configuration file.
- The *ATM Configuration and Management Manual* documents the Asynchronous Transfer Mode (ATM) product.
- The *DNS Configuration and Management Manual* provides information about the BIND 9.*x*-based DNS subsystem available for G06.25 and later RVUs and H06.03 and later RVUs.

X25AM Manuals

The X25AM subsystem manual set includes the following manuals:

- The *X25AM Configuration and Management Manual* describes how to install and configure the X25AM communications subsystem on NonStop S-series systems. This manual also describes the Subsystem Control Facility (SCF) for X25AM.
- The *X25AM Programming Manual* describes the programming requirements for developing an X25AM application.
- The *X25AM Management Programming Manual* describes the management-programming interfaces to the X25AM subsystem and how to use them.

How This Manual Is Organized

The following table summarizes the contents of this manual:

Section and Title	Description
Section 1, Configuration Quick Start	This section provides concise examples of setting up the HP NonStop TCP/IP environment. The services are not described in detail in this section; for more detailed information, see Networking Services Provided in the Guardian Environment on page 3-28, and Appendix B, NonStop TCP/IP Processes and Protocols. For more complex configuration examples, see Section 3, Configuring the NonStop TCP/IP Subsystem.
Section 2, Overview of NonStop TCP/IP	This section describes the NonStop TCP/IP subsystem in relation to other NonStop data communications subsystems.
Section 3, Configuring the NonStop TCP/IP Subsystem	This section discusses a set of command files that you can use to start and configure your NonStop TCP/IP environment over a network connected to a NonStop system. This section also provides information about networking services provided by NonStop TCP/IP and explains how to configure the OSS environment to use NonStop TCP/IP. See Appendix A, Configuration Reference, for reference information about the DNS server and the standard resource record format.
Section 4, SCF Reference	This section provides information about: ● The Subsystem Control Facility (SCF) ● SCF commands available for TCP/IP ● The PTrace facility
Appendix A, Configuration Reference	This appendix provides reference material required for configuring your NonStop TCP/IP subsystem.
Appendix B, NonStop TCP/IP Processes and Protocols	This appendix provides additional information about the NonStop server implementation of Transmission Control Protocol/ Internet Protocol. It also includes information on the services provided by the NonStop TCP/IP environment not documented in Section 1, Configuration Quick Start and Section 3, Configuring the NonStop TCP/IP Subsystem.

Section and Title	Description
Appendix C, Well-Known Port Numbers for TCP and UDP	This appendix lists the port numbers preassigned to specific services when accessed from TCP or UDP.
Appendix D, SCF Command Summary	This appendix summarizes the SCF command syntax.
Appendix E, SCF Error Messages	This appendix contains a description of the TCP/IP subsystem SCF error messages.
Appendix F, NonStop Systems Used as Internet Gateways	This appendix describes the use of a NonStop system as an internet gateway.
Glossary	Defines the terms used in this 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

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

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter 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; enter 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 may 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 may 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 entered 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 enter 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 the following example, there are no spaces 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

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

Bold Text. Bold text in an example indicates user input entered 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 might 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 might 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`*

Notation for Management Programming Interfaces

The following list summarizes the notation conventions used in the boxed descriptions of programmatic commands, event messages, and error lists in this manual.

UPPERCASE LETTERS. Uppercase letters indicate names from definition files; enter these names exactly as shown. For example:

```
ZCOM-TKN-SUBJ-SERV
```

lowercase letters. Words in lowercase letters are words that are part of the notation, including Data Definition Language (DDL) keywords. For example:

```
token-type
```

!r. The !r notation following a token or field name indicates that the token or field is required. For example:

```
ZCOM-TKN-OBJNAME      token-type ZSPI-TYP-STRING.          !r
```

!o. The !o notation following a token or field name indicates that the token or field is optional. For example:

```
ZSPI-TKN-MANAGER      token-type ZSPI-TYP-FNAME32.         !o
```

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.

HP Encourages Your Comments

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

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

1 Configuration Quick Start

This section provides concise examples of setting up the HP NonStop TCP/IP environment. The services are not described in detail in this section; for more detailed information, see Networking Services Provided in the Guardian Environment on page 3-28, and Appendix B, NonStop TCP/IP Processes and Protocols. For more complex configuration examples, see Section 3, Configuring the NonStop TCP/IP Subsystem.

The first example establishes a NonStop TCP/IP host environment which includes a TELNET service (TACL) and a LISTNER service (FTP server, ECHO server and FINGER server). Host name resolution is through a HOSTS file.

The second example shows how to safely stop your NonStop TCP/IP environment. Before you shut down the machine or bring up a new version of NonStop TCP/IP, follow this procedure.

The third example shows how to add an external Domain Name Server for address resolution. Many IP networks, particularly those connected to the Internet, include Domain Name Servers for host name resolution. In this example, the use of an external Domain Name Server is added for host name resolution.

The fourth example shows how to take advantage of NonStop parallel architecture by configuring multiple NonStop TCP/IP environments in multiple processors (CPUs). In this example, a second set of NonStop TCP/IP processes are configured against a second host IP address. A new command file called TCPIPDN1 is needed to shut down the second NonStop TCP/IP environment. The fifth example shows how to create the command file to bring down the second environment.

The sixth and seventh examples show you how to configure TCP/IP for Asynchronous Transfer Mode (ATM).

Gather Information

Before you begin to configure NonStop TCP/IP you’ll need to gather information about your system. Add the information you gather to the form in Figure 1-1, TCP/IP Configuration Form, on page 1-5.

Host Name

This is the name of the host being set up. The host name is arbitrary, you assign it. Example 1-1 uses the host name *breeze*. Example 1-6 uses the host name *TCPX*. This is item 1 in the configuration form.

IP Address of Host

This is the IP address of the host being set up. Get this IP address from the network administrator. Example 1-1 uses the IP address *150.20.30.1*. This is item 2 in the configuration form.

Name of a SLSA LIF

Follow the instructions below to obtain the name of a SLSA LIF.

1. Obtain a list of all LIFs by entering the following command at the SCF prompt:

   ```
   STATUS LIF $ZZLAN.*
   ```

2. Using one of the LIF names displayed, determine if the LIF is of TYPE ETHERNET by entering toe following command:

   ```
   INFO LIF $ZZLAN.lifname
   ```

3. Ensure that the LIF is accessible to all processors on which you plant to run NonStop TCP/IP by entering the following command:

   ```
   STATUS LIF $ZZLAN.lifname, DETAIL
   ```

   In the display, the field named “CPUs with Data Path” lists the available processors.

Example 1-1 uses the LIF name *LAN01*.

This is item 3 in the configuration form.

Gateway Router Address

The gateway router address for connection to non-local systems must be an IP address associated with a router on your network subnet. Get the gateway router address from the network administrator.

Example 1-1 uses the router address *150.20.30.2*.

This is item 4 in the configuration form.

Domain Name

The domain name is arbitrary, you assign it. Example 1-3 uses the domain name *mydomain.com*. This is item 5 in the configuration form.

Domain Name Server Address

Get the domain name server (DNS) address from the DNS administrator. Example 1-3 uses the domain name server address *mydomain.com*. This is item 6 in the configuration form.

IP Address of Second Host

Get the IP address of the second host from your network administrator. Example 1-4 uses the IP address *150.20.30.3*. This is item 7 in the configuration form.

Name of Second Host

You'll need this name when adding a second TCP/IP host to your system. This name is arbitrary, you assign it. Example 1-4 uses the host name *gust*. This is item 8 in the configuration form.

Name of Second SLSA LIF

Follow the instructions for choosing a SLSA LIF under the subheading "Name of SLSA LIF," above, and select a second SLSA LIF. Example 1-4 uses a SLSA LIF name *LAN02*. This is item 9 in the configuration form.

Opener Process Name

Enter the following command to list the opener processes:

```
LISTOPENS tcpip-process-name
```

Choose any of the open process names except the process that is running your home terminal.

To find the open process that is running your home terminal, follow the procedures below:

1. At the TACL prompt, enter the WHO command.
2. In the WHO display look for the home terminal listing near the top of the display and make a note of it. (Note just the portion of the name following the dollar sign ($).)
3. At the SCF prompt enter the LISTDEV TCPIP command.
4. Issue a LISTOPENS PROCESS $*process-name* command against each of the processes listed in the display for the LISTDEV TCPIP command until you find the process that is running the TACL prompt of your home terminal. (This is the name you noted in step 2.)

Example 1-5uses opener process names $*ZTN1* and $*LSN1*.

This is item 10 in the configuration form.

TCP/IP Process Name

The TCP/IP process name is arbitrary, you assign it. Example 1-6 uses the name $*ZTC1*. This item 11 in the configuration form.

SUBNET Name

The SUBNET names are arbitrary, you assign them. Example 1-6 uses the SUBNET names #*EN1* and *LO**. This is item 12 in the configuration form.

IP Address of the SUBNET

Get the SUBNET IP address from the network administrator. Example 1-6 uses the address *172.16.192.203*. This is item 13 in the configuration form.

DEVICENAME for the Adapter

To choose a DEVICENAME for an adapter, issue the following command at the SCF prompt:

```
NAMES ADAPTER $ZZATM.*
```

This command results in a listing of the operational ATM adapters on the system. Choose an adapter and use its name as the DEVICENAME for the adapter.

Example 1-6 uses the DEVICENAME $*AM2*.

This is item 14 in the configuration form.

Server Name

Server names are arbitrary, you assign them. Example 1-6 uses the server names #*SRV1* and #*SRV2*. This is item 15 in the configuration form.

Server ATM Address

Get the server ATM addresses from the network administrator. Example 1-6 uses the following server ATM addresses:

```
H47000580FFE1000000F21A29EB0000000001A500
H47000580FFE1000000F21A29EB000000000A300
```

This is item 16 in the configuration form.

Subnet Mask Used on Local Subnet

The subnet mask used on the local subnet depends on the IP class and your network set up. Get this mask from the network administrator. Example 1-6 uses the subnet mask *255.255.255.0* in dot notation (%*hffffff00* in hexadecimal format). This is item 17 in the configuration form.

Route Name

The route name is arbitrary, you assign it. Example 1-6 uses the route name #*DEF*. (Route name #RT is reserved.) This is item 18 in the configuration form.

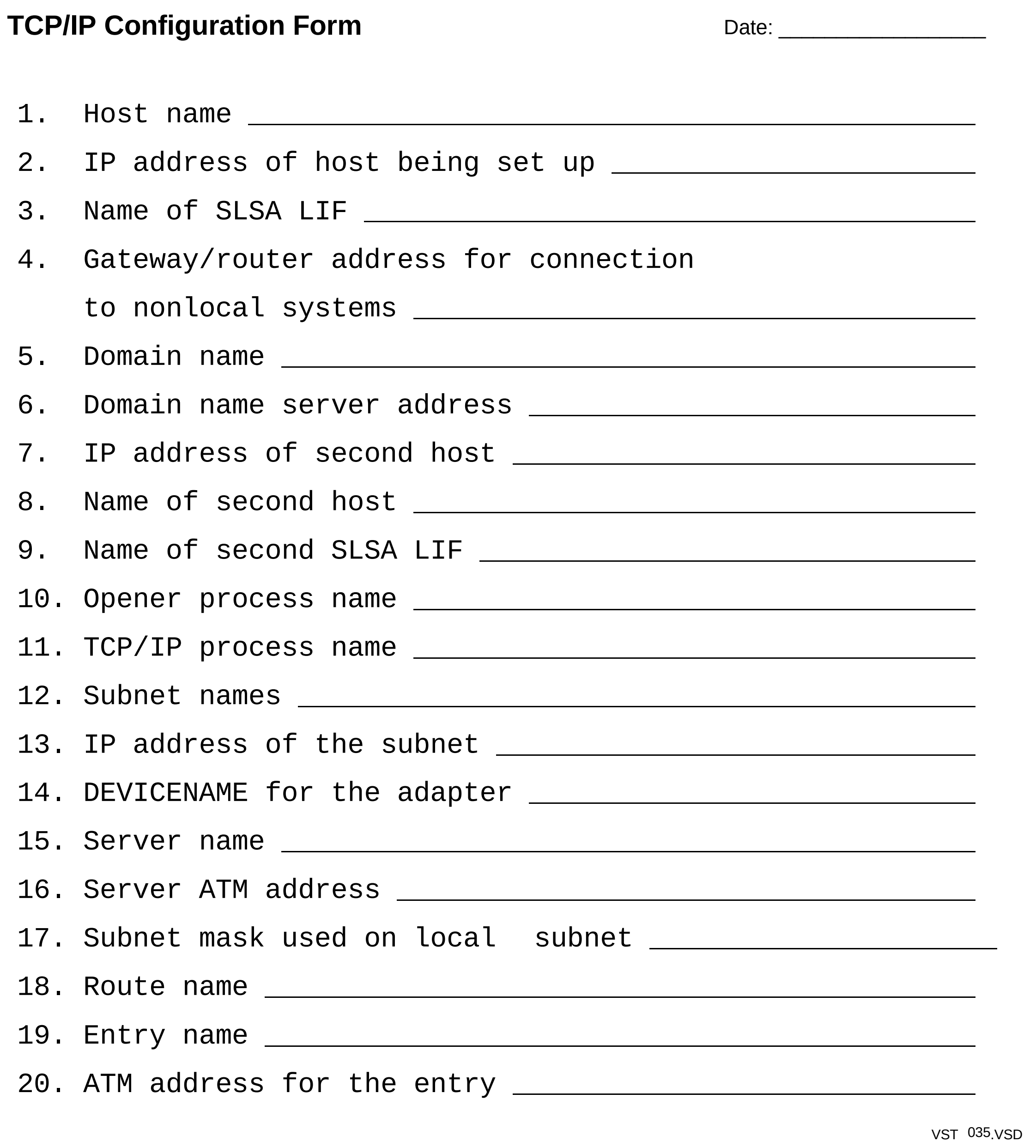
Entry Name

The entry name is arbitrary, you assign it. Example 1-6 uses the entry name #*A2*. This is item 19 in the configuration form.

ATM Address of the Entry

Get entry ATM address from the network administrator. Example 1-6 uses the address *%H47000580FFE1000000F21A29EB000000000A100*.

Figure 1-1. TCP/IP Configuration Form



TCP/IP Configuration Form Date: ________________

```
1.  Host name ____________________________________________
2.  IP address of host being set up ______________________
3.  Name of SLSA LIF _____________________________________
4.  Gateway/router address for connection
    to nonlocal systems __________________________________
5.  Domain name __________________________________________
6.  Domain name server address ___________________________
7.  IP address of second host ____________________________
8.  Name of second host __________________________________
9.  Name of second SLSA LIF ______________________________
10. Opener process name __________________________________
11. TCP/IP process name __________________________________
12. Subnet names _________________________________________
13. IP address of the subnet _____________________________
14. DEVICENAME for the adapter ___________________________
15. Server name __________________________________________
16. Server ATM address ___________________________________
17. Subnet mask used on local  subnet _____________________
18. Route name ___________________________________________
19. Entry name ___________________________________________
20. ATM address for the entry ____________________________
```

VST 035.VSD

Task Summary: Host Environment With TELNET and LISTNER

Task 1: Edit the HOSTS file

Task 2: Create a command file

Task 3: Start the NonStop TCP/IP Environment

Assumptions

- The SCF environment is operational (that is, SCP is running)
- QIOMON is running
- The SLSA subsystem is configured (see the *LAN Configuration and Management Manual* to configure the SLSA subsystem)

You need to substitute real values for the following values; (these variables are indicated by italics in the example):

- IP Address of Host being set up (the example uses *150.20.30.1*)
- Host Name of host being set up (the example uses *breeze*)
- Subnet Mask Used on Local Subnet (the example uses *255.255.255.0* in dot notation, which is *%hffffff00* in hexadecimal format)
- Name of a SLSA LIF (the example uses *LAN01*). See the Hint below for choosing an appropriate LIF.
- Gateway Router Address for connection to non-local systems (the example uses *150.20.30.2*)

Tasks

Hint. Before you add your NonStop TCP/IP process pair, issue an SCF STATUS LIF $ZZLAN.* command. Select a running LIF and issue a STATUS LIF $ZZLAN.*lif-name* on that LIF. The "CPUs with Data Path" field lists the available CPUs. Select two CPUs in the "CPUs with Data Path" list. Use the CPUs you selected when specifying the primary and backup CPU numbers for the TCP/IP process. In addition, use the same CPUs when adding TELNET and LISTNER. When you add the SUBNET (see ADD SUBNET Command on page 4-21 for details) use the LIF name that you just chose for the DEVICENAME. (These items are highlighted in the command files shown in Example 1-1, Example 1-3, and Example 1-4.)

```
>SCF
->STATUS LIF $ZZLAN.*

SLSA Status LIF
Name                 State         Access State
$ZZLAN.LANY          STARTED            UP
$ZZLAN.LANX          STARTED            UP
$ZZLAN.TOK20A        STARTED            UP
$ZZLAN.TOK10A        STARTED            UP
$ZZLAN.LAN21         STARTED            UP
```

The following example shows the display that results from the STATUS LIF $ZZLAN.<*lif-name*> command:

```
-> STATUS LIF $ZZLAN.LAN21, detail

SLSA Detailed Status LIF \SYSTEM.$ZZLAN.LAN21

  Access State............... UP
  CPUs with Data Path........ ( 0 , 1, 2, 3)
  Potential Access CPUs...... ( 0, 1, 2, 3))
  State...................... STARTED
  Trace Filename.............
  Trace Status...............
```

1. Edit the $SYSTEM.ZTCPIP.HOSTS file to include the following entries:

```
127.0.0.1 localhost loopback
150.20.30.1 breeze
==150.20.30.1 is the IP Address of Host
==breeze is the Host Name
```

2. Create a TACL command file called TCPIPUP as in Example 1-1. (Note that the TCPIP RUN command does not specify priority (PRI). TCP/IP starts at a priority of 200 regardless of any priority specified for startup.)

Example 1-1. TCPIPUP Command File

```
 TCPIP/NAME $ZTC0,TERM $TRM0.#A,OUT $TRM0.#A,CPU 0,NOWAIT/1
 SCF/INLINE/
 INLPREFIX +
 + ASSUME PROCESS $ZTC0
 + ALTER,HOSTNAME "breeze"
==breeze is the Host Name
 + ALTER,HOSTID 150.20.30.1
==150.20.30.1 is the IP Address of Host
 + ALTER SUBNET #LOOP0,ADDRESS 127.1
 + ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN21, &
 IPADDRESS 150.20.30.1, IRDP OFF;
==LAN21 is the Name of a SLSA LIF
 + ALTER SUBNET #SN0,SUBNETMASK %HFFFFFF00
 + ADD ROUTE #ROUTE0,DESTINATION 0.0.0.0,GATEWAY 150.20.30.2
==150.20.30.2  is the Gateway Router Address
 + START SUBNET *
 + START ROUTE *
 INLEOF
 LISTNER/TERM $ZHOME,NAME $LSN0,CPU 0,NOWAIT,&
 PRI 160/$SYSTEM.ZTCPIP.PORTCONF
 TELSERV/TERM $ZHOME,NAME $ZTN0,CPU 0,NOWAIT,&
 PRI 170/ -BACKUPCPU 1
```

3. Issue the OBEY command on the TCPIPUP command file when running as user SUPER.SUPER to start the NonStop TCP/IP environment.

   ```
   >OBEY TCPIPUP
   ```

Hint. For NonStop TCP/IP applications such as FTP and TELNET to use the HOSTS file for name resolution, you must specify the following define. (Typically, this define is placed in the $SYSTEM.SYSTEM.TACLLOCL file so that all TACL users inherit the define):

```
ADD DEFINE =TCPIP^HOST^FILE,CLASS MAP,FILE &
$SYSTEM.ZTCPIP.HOSTS
```

Once this NonStop TCP/IP environment is running, you can use FTP to make transfers to or from this host. The TELSERV configuration, by default, gives users access to a TACL service.

Task Summary: Shutting Down TCP/IP

Task 1: Create a TACL command file

Task 2: Issue the OBEY command to stop the NonStop TCP/IP environment

Assumptions

There is a NonStop TCP/IP environment running on your system.

Tasks

1. Create a TACL command file as shown in Example 1-2.

Example 1-2. TCPIPDN Command File

```
SCF/INLINE/
INLPREFIX +
+ ABORT WINDOW $ZTN0.*
INLEOF
STOP $ZTN0
STOP $LSN0
SCF ABORT PROCESS $ZTC0
```

2. Issue the OBEY command on the TCPIPDN command file while running as user SUPER.SUPER.

```
>OBEY TCPIPDN
```

Task Summary: Adding an External Domain Name Server

Task 1: Edit the RESCONF file

Task 2: Delete the define for TCPIP^HOST^FILE from the TCPIPUP file

Task 3: Change the hostname in the TCPIPUP file

Task 4: Issue the OBEY command on the file

Assumptions

You need to substitute real values for the following configuration items for the values in this example; (these variables are indicated by italics in the example):

- Domain Name (example uses *mydomain.com*)
- Domain Name Server Address (DNS address) (example uses *150.20.30.10*)

Tasks

1. Edit the $SYSTEM.ZTCPIP.RESCONF file to include the following entries:

```
domain mydomain.com
nameserver 150.20.30.10
```

2. To ensure that the resolver routines use the DNS, ensure the define for TCPIP^HOST^FILE is not used by deleting the entry in the TCPIPUP and TACLLOCL files from the previous example.

3. Change the TCPIPUP command file to include the correct hostname for this system (including the domain suffix). Example 1-3 shows the new TCPIPUP command file:

Example 1-3. TCPIPUP Command File

```
 TCPIP/NAME $ZTC0,TERM $TRM0.#A,OUT $TRM0.#A,CPU 0,NOWAIT/1
 SCF/INLINE/
 INLPREFIX +
 + ASSUME PROCESS $ZTC0
 + ALTER,HOSTNAME "breeze.mydomain.com"
==breeze is the Host Name
 + ALTER,HOSTID 150.20.30.1
==150.20.30.1 is the Domain Name Server Address
 + ALTER SUBNET #LOOP0,ADDRESS 127.1
 + ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN21,&
 IPADDRESS 150.20.30.1, IRDP OFF
==LAN01 is the Name of a SLSA LIF
 + ALTER SUBNET #SN0,SUBNETMASK %HFFFFFF00
==%HFFFFFF00 is the Subnet Mask Used on Local Subnet
 + ADD ROUTE #ROUTE0,DESTINATION 0.0.0.0,GATEWAY 150.20.30.2
==150.20.30.2 is the Gateway Router Address
 + START SUBNET *
 + START ROUTE *
 INLEOF
 LISTNER/TERM $ZHOME,NAME $LSN0,CPU 0,NOWAIT,&
 PRI 160/ $SYSTEM.ZTCPIP.PORTCONF
 TELSERV/TERM $ZHOME,NAME $ZTN0,CPU 0,NOWAIT,&
 PRI 170/ -BACKUPCPU 1
```

4. Issue the OBEY command on the TCPIPUP command file while running as user SUPER.SUPER.

   ```
   >OBEY TCPIPUP
   ```

The TCPIPDN file does not change from the previous example.

Task Summary: Adding a Second TCP/IP Host Using a Second TCP/IP Process

Task 1: Create a TACL command file

Task 2: Issue the OBEY command on the file

Assumptions

You need to substitute real values for the following configuration items for the values in this example (these variables are indicated by italics in the example):

- IP Address of Second Host (the example uses *150.20.30.3*)
- Name of Second Host (the example uses *gust*)

- Name of Second SLSA LIF (the example uses *LAN02*). See Hint on page 1-7 for choosing an appropriate LIF.

Tasks

1. Create a new TACL command file called TCPIPUP1 to start the second NonStop TCP/IP environment using the contents of Example 1-4. The define for TCPIP^PROCESS^NAME is needed when starting applications which don't use the default NonStop TCP/IP process. The default NonStop TCP/IP process name is $ZTC0, so you do not need to use the define when using $ZTC0. When adding any further NonStop TCP/IP processes, use the define to ensure that applications use the correct NonStop TCP/IP process.

Example 1-4. TCPIPUP1 Command File

```
 TCPIP/NAME $ZTC1,TERM $TRM0.#A,OUT $TRM0.#A,CPU 1,NOWAIT/2
 SCF/INLINE/
 INLPREFIX +
 + ASSUME PROCESS $ZTC1
 + ALTER,HOSTNAME "gust.mydomain.com"
==gust is the Host Name
 + ALTER,HOSTID 150.20.30.3
==150.20.30.3 is the Name of Second Host
 + ALTER SUBNET #LOOP0,ADDRESS 127.1
 + ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN02,&
 IPADDRESS 150.20.30.2, IRDP OFF
==LAN02 is the Name of Second SLSA LIF
==150.20.30.2 is the Gateway Router Address
 + ALTER SUBNET #SN0,SUBNETMASK %HFFFFFF00
== %HFFFFFF00 is the Subnet Mask Used on Local Subnet
 + ADD ROUTE #ROUTE0,DESTINATION 0.0.0.0,GATEWAY 150.20.30.2
 + START SUBNET *
 + START ROUTE *
 INLEOF
 ADD DEFINE =TCPIP^PROCESS^NAME,CLASS MAP,FILE $ZTC1
 LISTNER/TERM $ZHOME,NAME $LSN1,CPU 1,NOWAIT,&
 PRI 160/$SYSTEM.ZTCPIP.PORTCONF
 PARAM TCPIP^PROCESS^NAME $ZTC1
 TELSERV/TERM $ZHOME,NAME $ZTN1,CPU 1,NOWAIT,&
 PRI 170/ -BACKUPCPU 2
```

2. Issue the OBEY command on the TCPIPUP1 command file when running as user SUPER.SUPER to start the second NonStop TCP/IP environment.

   ```
   >OBEY TCPIPUP1
   ```

Task Summary: Shutting Down the Second TCP/IP Environment

Task 1: Create a TACL command file

Task 2: Issue an OBEY command on the file

Assumptions

- A second TCP/IP process is running on the system.
- You need to substitute real values for the opener process names. (List all openers of the TCP/IP home terminal process provider.)

Tasks

1. Create a TACL command file with the contents shown in Example 1-5.

Example 1-5. TCPIPDN1 Command File

```
 SCF/INLINE/
 INLPREFIX +
 + ABORT WINDOW $ZTN1.*
 INLEOF
 STOP $ZTN1
==$ZTN1 is an Opener Process Name
 STOP $LSN1
==$LSN1 is an Opener Process Name
 SCF ABORT PROCESS $ZTC1
```

2. Issue an OBEY command on the TCPIPDN1 file:

```
>OBEY TCPIPDN1
```

Task Summary: ATM ARP Server Client

Task 1: Create a TACL command file

Task 2: Issue an OBEY command on the file

Assumptions

- The SCF environment is operational (that is, SCP is running)
- QIOMON is running
- The ATM subsystem is configured (see the *ATM Configuration and Management Manual* to configure the ATM subsystem)
- You must have the correct ATM address (see your network manager)

You need to substitute real values for the following variables; (these variables are indicated by italics in the example):

- Process name (the example uses $*ZTC1*)

Hint. Before you add your NonStop TCP/IP process pair, issue an SCF NAMES ADAPTER ZZATM.* command. Select a running ATM adapter and issue an INFO ADAPTER $*adapter-name*. The “Access List” field lists the available CPUs. Select two CPUs from the “CPUs with Data Path” list. Use the CPUs you selected when specifying the primary and backup CPU numbers for the TCP/IP process. When you add the subnet (see ADD SUBNET Command on page 4-21 for details) use your chosen adapter name for the DEVICENAME. (The adapter names are highlighted in the command files shown in Example 1-6, and Example 1-7.)

```
>SCF
->NAMES ADAPTER $ZZATM.*
This command returns a list of operational ATM adapters on
the system. Choose an adapter, and issue the INFO ADAPTER
$adapter-name command:

-> INFO ADAPTER $AM2
```

The following example shows the display that results from the INFO ADAPTER command:

```
ATM Info ADAPTER $AM2

Location (grp,mod,slot).... 1 ,1 ,54
Access List....................  ( 2,3 )
Amp Filename (in use).....  \SYSTEM.SYS02.AMP
Amp Filename (config)......  $SYSTEM.SYSNN.AMP
LIB Filename (in use)......... $SYSTEM.SYS02.ZATMSRL
LIB Filename (config).........  $SYSTEM.SYSNN.ZATMSRL
```

- SUBNET Name (the example uses #*EN1* and #*LO**)
- IP Address of the SUBNET (the example uses *172.16.192.203*)
- DEVICENAME for the Adapter (the example uses $*AM2*). See the Hint on page 1-13 for information on how to obtain the adapter/device name.
- Server Name (the example uses #*SRV1* and #*SRV2*)
- Server ATM Address (the example uses *H47000580FFE1000000F21A29EB0000000001A500* and *H47000580FFE1000000F21A29EB0000000001A300*)
- Subnet Mask Used on Local Subnet (the example uses *255.255.255.0* in dot notation, which is %*hffffff00* in hexadecimal format)
- Route Name (the example uses #*DEF* (remember #RT is reserved))
- Gateway Router Address address for connection to non-local systems (the example uses *172.16.192.1*)
- Host Name (the example uses “*TCPX*”)
- Entry Name (the example uses #*A2*)
- ATM Address of the Entry (for the Gateway) %*H47000580FFE1000000F21A29EB0000000001A100*)

Tasks

1. Create a TACL command file with the contents shown in Example 1-6.

Example 1-6. ATMUP Command File

```
TCPIP/NAME $ZTC1,TERM $TRM0.#A,OUT $TRM0.#A,CPU 1,NOWAIT/2
SCF/INLINE/
INLPREFIX+
+ALLOW 100 ERRORS
+ADD SUBNET #EN1,TYPE ATM,IPADDRESS 172.16.192.203, &
     DEVICENAME $AM2
==#EN1 is the SUBNET Name
==172.16.192.203 is the IP Address of the SUBNET
==$AM2 is the DEVICENAME for the Adapter
+ADD SERVER #SRV1,SUBNET “#EN1”  &

,ATMADDR(ISONSAP:%H47000580FFE1000000F21A29EB0000000001A500)
==ISONSAP address is a Server ATM Address
==#SVR1 is a Server Name
+ADD SERVER #SRV2,SUBNET “#EN1”  &

,ATMADDR(ISONSAP:%H47000580FFE1000000F21A29EB0000000001A300)
+ALTER SUBNET #EN1, SUBNETMASK %HFFFFFF00
==%HFFFFFF00 is a Server ATM Address
+ALTER SUBNET #LOOP0,IPADDRESS 127.1
+START SUBNET *
+ADD ROUTE #DEF, DESTINATION 0.0.0.0, GATEWAY 172.16.192.1
==172.16.192.1 is the Gateway Router Address
+ALTER,HOSTNAME “TCPX”
==TCPX is the Host Name
+START ROUTE *
+ALTER,TCPSENDSPACE 12288
+ALTER,TCPRECVSPACE 12288
+ADD ENTRY #A2,IPADDRESS  172.16.192.1,TYPE ATMARP  &

,ATMADDR(ISONSAP:%H47000580FFE1000000F21A29EB0000000001A100)
==#A2 is the Entry Name
INLEOF
```

2. Issue an OBEY command on the ATMUP command file.

```
>OBEY ATMUP
```

Task Summary: TCP/IP Process Configured as an ARP Server

Task 1: Create a TACL command file.

Task 2: Issue an OBEY command on the file.

Assumptions

- The SCF environment is operational (that is, SCP is running)
- QIOMON is running
- The ATM subsystem is configured (see the *ATM Configuration and Management Manual* to configure the ATM subsystem)
- You must have the correct ATM address (see your network manager)

You need to substitute real values for the following variables (indicated by italics in the example):

- TCP/IP Process Name (the example uses $*ZTC1*). See the Hint on page 1-13 for information on how to choose the primary and backup CPUs.
- SUBNET Name (the example uses #*EN1*)
- IP Address of the SUBNET (the example uses *172.16.192.203*)
- DEVICENAME for the Adapter (the example uses $*AM2*). See the Hint on page 1-13 for information on how to choose the adapter/device name.
- Subnet Mask Used on Local Subnet (the example uses *255.255.255.0* in dot notation, which is %*hffffff00* in hexadecimal format)
- Host Name (the example uses "*TCPX*")
- Route Name (the example uses #*DEF* (remember #RO U is reserved))
- Gateway Router Address for connection to non-local systems (the example uses *172.16.192.1*)
- Entry Name (the example uses #*A2*)
- ATM Address of the Entry (the example uses %*H47000580FFE1000000F21A29EB0000000001A100*)

Tasks

1. Create a TACL command file with the contents shown in Example 1-6.

Example 1-7. ATMUP2 Command File

```
TCPIP/NAME $ZTC1,TERM $TRM0.#A,OUT $TRM0.#A,CPU 1,NOWAIT/2
==$ZTC1 is the TCP/IP Process Name
SCF/INLINE/
INLPREFIX +
+ALLOW 100 ERRORS
+ADD SUBNET #EN1,TYPE ATM,IPADDRESS 172.16.192.203, &
     DEVICENAME $AM2, ARPSERVER ON
==#EN1 is the SUBNET Name
==172.16.192.203 is the IP Address of the SUBNET
==$AM2 is the DEVICENAME for the Adapter
+ALTER SUBNET #EN1, SUBNETMASK %HFFFFFF00
==%HFFFFFF00 is the Subnet Mask Used on Local Subnet
+ALTER SUBNET #LOOP0,IPADDRESS 127.1
+START SUBNET *
+ADD ROUTE #DEF, DESTINATION 0.0.0.0, GATEWAY 172.16.192.1
==#DEF is the Route Name
+ALTER,HOSTNAME “TCPX”
==TCPX is the Host Name
+START ROUTE *
+ALTER,TCPSENDSPACE 12288
+ALTER,TCPRECVSPACE 12288
+ADD ENTRY #A2,IPADDRESS  172.16.192.1, &
    TYPE ATMARP &
   ,ATMADDR(ISONSAP:%H47000580FFE1000000F21A29EB0000000001A100)
==#A2 is the Entry Name
==the ISONSAP address is the ATM Address of the Entry
INLEOF
```

2. Issue an OBEY command on the ATMUP2 command file.

   ```
   >OBEY ATM2UP
   ```

2 Overview of NonStop TCP/IP

This section describes the NonStop TCP/IP subsystem in relation to other NonStop data communications subsystems.

Note. Only Ethernet communications through the G4SA are available on the Integrity NonStop server; to use other protocols and adapters, the Integrity NonStop server must be connected to an IOMF2 customer replaceable unit (CRU) in a NonStop S-series I/O enclosure. For more information, see the *Introduction to Networking for HP Integrity NS-Series Servers*.

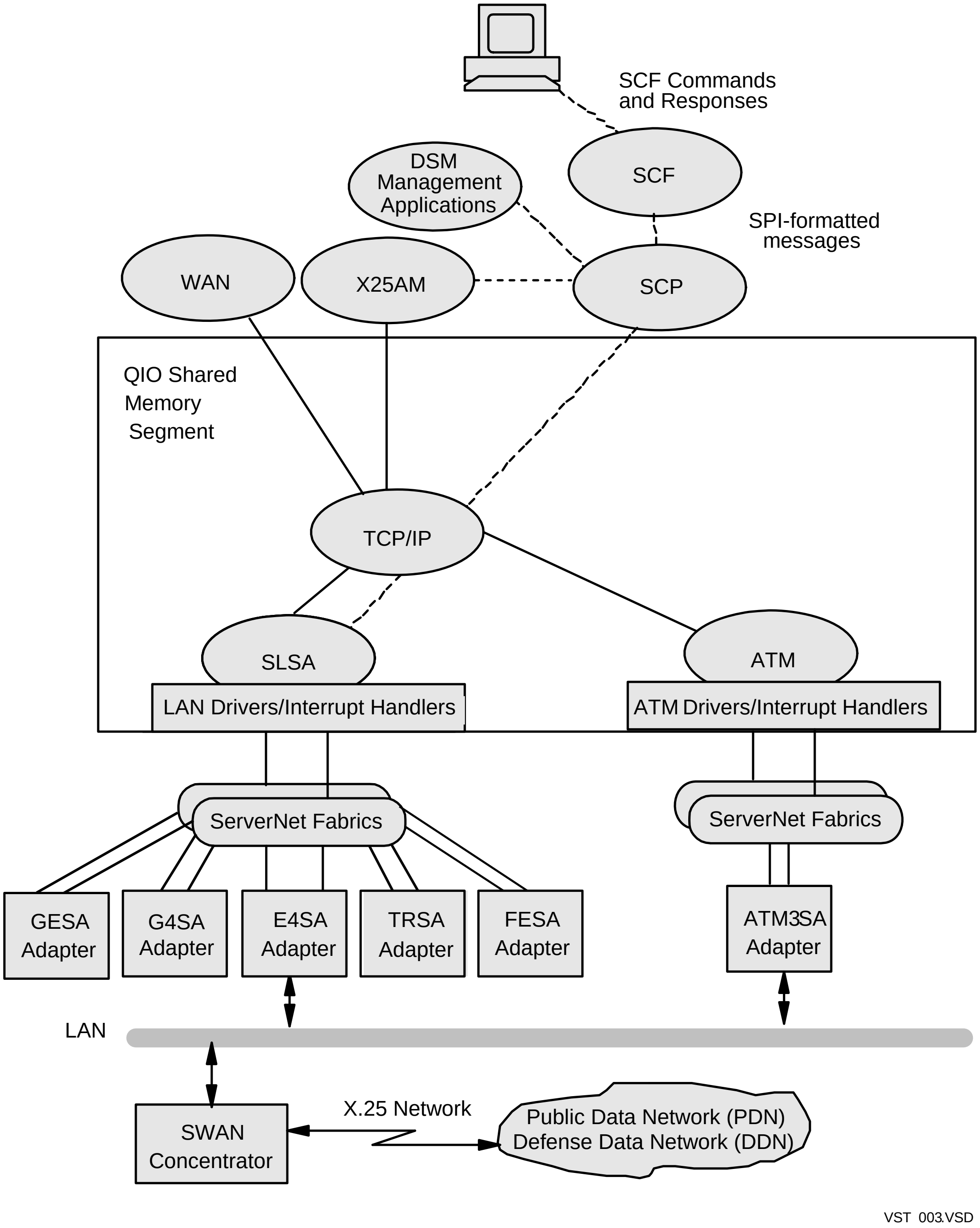
NonStop TCP/IP provides access to a network, allowing you to use FTP to transfer files to your NonStop system, and run web-based services through facilities described in Appendix B, NonStop TCP/IP Processes and Protocols.

NonStop TCP/IP runs as a process pair on the HP NonStop operating system. It allows communication between heterogeneous systems in a multi network environment.

NonStop TCP/IP software conforms to a number of Request for Comments (RFCs), Internet Engineering Notes (IENs), and Military Standards (MIL-STDs) maintained by the Defense Data Network (DDN) at the DDN Network Information Center (NIC) operated by SRI International in Menlo Park, California. The RFCs, IENs, and MIL-STDs define the protocols implemented by the NonStop TCP/IP software. Conformance to DDN specifications includes operation of the Internet Protocol (IP) over Ethernet LANs.

Figure 2-1 shows a high-level view of the NonStop TCP/IP subsystem and its relationship to the QIO, SLSA, ATM and X25AM subsystems, and to various LAN, WAN and ATM adapters. Figure 2-1 also shows the management interfaces involved in running TCP/IP. Descriptions of the system components shown in Figure 2-1 follow the figure.

Figure 2-1. NonStop TCP/IP Subsystem Within the NonStop System



Management Interfaces and Network Components Shown in Figure 2-1

The following subsections describe the management interfaces and network components shown in Figure 2-1.

The NonStop TCP/IP product for NonStop servers provides transparent connections to IEEE 802.3 Ethernet LANs and IEEE 802.5 token-ring LANs through the SLSA subsystem. NonStop TCP/IP uses the WAN subsystem to interface to X.25 packet-switched networks. For more information about the WAN subsystem, refer to the *WAN Subsystem Configuration and Management Manual*. NonStop TCP/IP uses the ATM subsystem to interface to ATM networks. For more information about the ATM subsystem, refer to the *ATM Configuration and Management Manual*.

ATM3SA

A ServerNet adapter that provides access to Asynchronous Transfer Mode (ATM) networks from a NonStop S-series server. The ATM3SA supports the ATM User-Network Interface (UNI) specification over a 155-megabit per second (Mbps) OC-3 Sonet (Synchronous Optical Network) connection.

DSM

An interface and set of tools used to configure, control, and manage NonStop servers and Expand networks. DSM includes the SPI interface-to-management process, in addition to other tools that assist in the development of applications.

E4SA

A ServerNet adapter for Ethernet local area networks (LANs) that contains four Ethernet ports.

Fast Ethernet ServerNet Adapter (FESA)

A single-port ServerNet adapter that supports 100-Mbps and 10-Mbps Ethernet data-transfer rates on a NonStop S-series server. The FESA installs into an Ethernet port. One FESA is supported for each system enclosure.

G4SA

A multiport ServerNet adapter that provides 1000 megabits/second (Mbps) data transfer rates between NonStop S-series servers, Integrity NonStop NS-series servers, and Ethernet LANs. The G4SA is the only LAN adapter supported for the I/O Adapter Module (IOAM) enclosure, and it is installed in slots 1, 2, 3, 4, and 5 of an IOAM. Although the G4SA supersedes the Ethernet 4 ServerNet adapter (E4SA), Fast Ethernet ServerNet adapter (FESA), and the Gigabit Ethernet ServerNet adapter (GESA), it cannot be installed in an HP NonStop S-series I/O enclosure.

GESA

A single-port ServerNet adapter that provides 1000 Mbps data transfer rates between NonStop S-series systems and Ethernet LANs.

SCF and SCP

SCF is an interactive interface that allows operators and system managers to configure, control, and monitor the NonStop TCP/IP subsystem. SCF is part of DSM. The Subsystem Control Point (SCP) provides an interface to the I/O processes of the various subsystems.

SLSA

A subsystem of the NonStop operating system for configuration and management of ServerNet LAN objects in G-series RVUs.

SWAN Concentrator

A wide-area network data communications peripheral that provides connectivity to a NonStop S-series server or Integrity NonStop NS-series server.

TRSA

A ServerNet adapter that provides a single line from a NonStop S-series server to a token-ring network, allowing the server to act as a station on the ring. Speeds are switchable between 4 megabytes (MB) and 16 MB, and the media can be either shielded twisted pair (STP) or unshielded twisted pair (UTP).

X25AM

For WANs, a product that implements the services of the Network Layer and layers below.

QIO

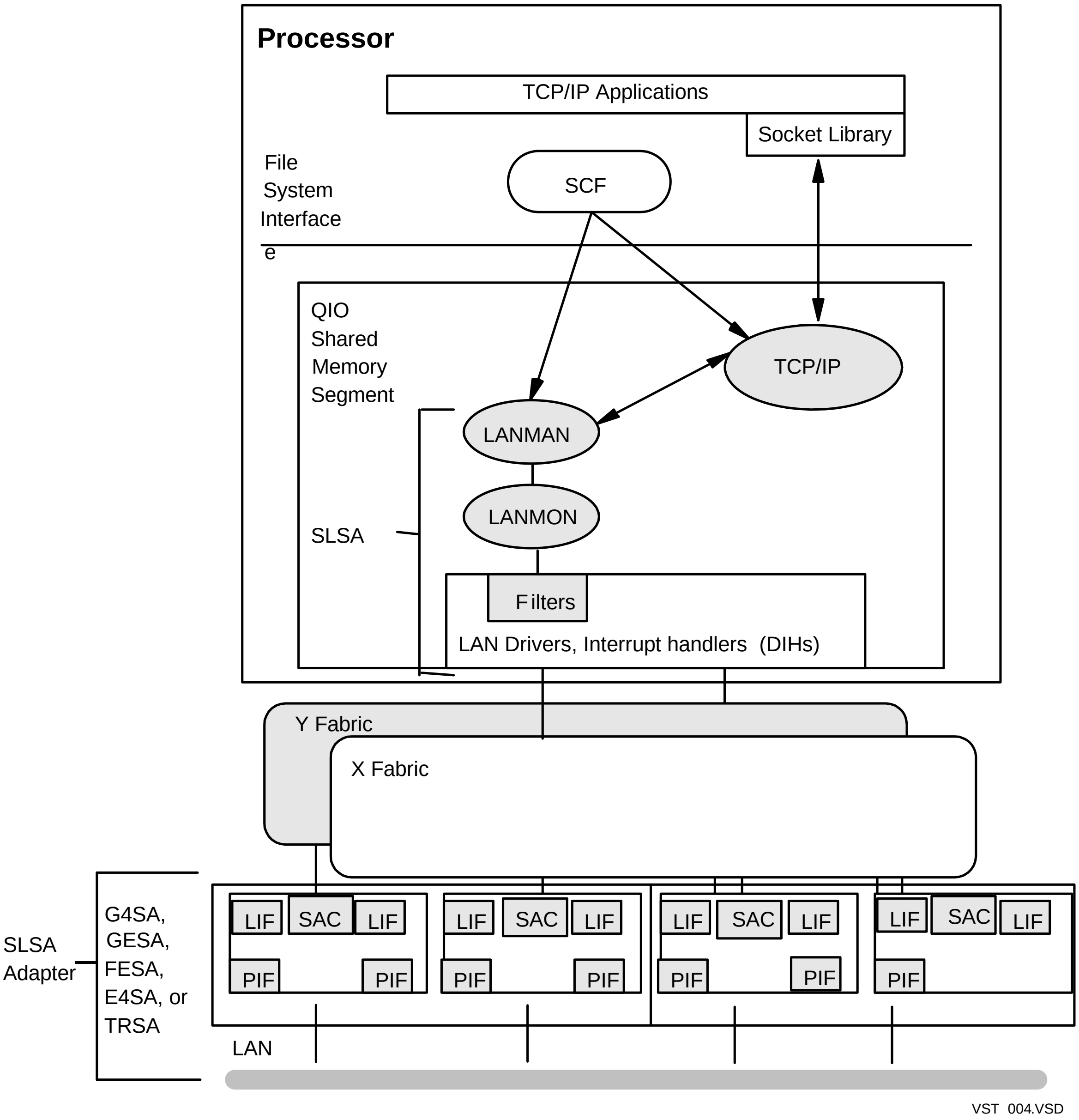
The QIO subsystem improves input/output (I/O) performance for NonStop TCP/IP by providing more efficient sharing of processor memory.

For more information about QIO, see the *QIO Configuration and Management Manual*.

NonStop TCP/IP Interface to LAN, WAN, and ATM networks

The NonStop TCP/IP process interfaces to the TRSAs, E4SAs, FESAs, GESAs, and G4SAs through the SLSA subsystem. As shown in Figure 2-2, the SLSA subsystem provides logical interfaces (LIFs) through which packets pass to filters. Packets pass through the rest of the SLSA subsystem, which consists of ServerNet Addressable Adapters (SACs) and PIFs, then out to a LAN, WAN, or X.25 connection. A brief description of the SLSA components relevant to this discussion follows Figure 2-2. The NonStop TCP/IP process interfaces to ATM adapters (ATM3SAs) through the ATM subsystem which is not shown in Figure 2-2 (see Figure 2-1).

Figure 2-2. TCP/IP Relationship to the SLSA Subsystem



Management Interfaces and Network Components Shown in Figure 2-2

Filters

Filters provide a logical mechanism whereby frames received from the LAN can be sorted, then delivered, to a particular client such as NonStop TCP/IP. Filters replace the PORT objects used in systems prior to the ServerNet architecture in the sense that filters are the final destination for data received from the LAN. The NonStop TCP/IP process in NonStop S-series systems registers ARP filters and IP filters through a LIF.

LANMAN

The LANMAN process is part of the SLSA subsystem. The LANMAN process performs the following tasks:

- Starts and manages the SLSA subsystem objects and the LAN monitor (LANMON) process
- Assigns ownership of Ethernet adapters to the LANMON processes in the system

Subsystem Control Facility (SCF) commands are directed to the LANMON processes for configuring and managing the SLSA subsystem and the Ethernet adapters.

LANMON

The LANMON process is part of the SLSA subsystem. The LANMON process has ownership of the Ethernet adapters controlled by the SLSA subsystem.

SLSA Adapters

For more information about SLSA adapters, see the manuals listed under SLSA Adapters Manuals on page xiv.

Overview of TCP/IP over ATM

NonStop TCP/IP includes support for TCP/IP over ATM networks. ATM stands for Asynchronous Transfer Mode, a networking technology that relays fixed-sized cells as opposed to frames or packets. These fixed-sized cells reduce processing time in the switching hardware giving ATM high bandwidth networking services. ATM can transmit data, voice, and video information on both local and wide area networks.

HP provides ATM technology through the ATM subsystem, which is described in detail in the *ATM Configuration and Management Manual*.

The rest of this subsection provides high-level information about TCP/IP for ATM.

ATM Term Definitions

The following terms are used throughout this manual in reference to ATM.

ATMARP Table

The TCP/IP process maintains a table of ATM IP addresses in an ATMARP table. Similar to the ARP table TCP/IP uses to map an IP address to a MAC address in Ethernet and token-ring networks, TCP/IP uses the ATMARP table to map IP addresses to an ATM address in ATM networks. You can view and change the ATMARP table.

ATMARP Server

An ATMARP server is a process used by other ATM endpoints to resolve IP to ATM addresses.

ATM Address Table

The ATM address table stores current Permanent Virtual Circuit (PVC) and Switched Virtual Circuit (SVC) connection information. See PVC and SVC below. You can view, but not change, the ATM address table.

Endpoint

Endpoint has two definitions:

- A unique 32-bit value assigned to an ATM address entry
- A device on a network

PVC

A PVC is a logical connection between network entities permanently maintained by the ATM network. NonStop TCP/IP does not support a PVC connection to an ARP server.

SVC

An SVC is a logical connection between network entities established for the duration of the communication and then destroyed.

RFC Compliance

The default IP MTU size for ATM type SUBNETs is 9180 bytes. The TCP/IP process supports IP Path MTU discovery (RFC-1191) on TCP connections but does not support Path MTU discovery for UDP type sockets.

The following are some possible differences between the IETF standards and NonStop TCP/IP:

- Path MTU Discovery is not supported on UDP-type sockets.
- PVCs must be added specifying the IP address of the other endpoint.
- MIB is not supported.
- ARP table entries associated with PVCs are not aged.
- The retry-timeout value for connection attempts to an ATMARP server is 10 seconds. Maximum retry count is 5 before attempting another server if more than one is configured.
- Only 20 octet (40 digit) NSAP addressing is supported. ATM Subaddresses are not supported.
- The ATM subsystem does not negotiate the AAL CPCS-SDU size using the ATM signalling protocol.

The TCP/IP process supports both SVCs and PVCs. (NonStop TCP/IP does not support a PVC connection to an ARP server.) Each ATM type SUBNET can be configured to operate as an ATMARP server.

The TCP/IP process is compatible with IP over ATM networks that are based on the older RFC-1577 document. When used as an ATMARP server, clients that are RFC-1577 based are also supported.

Hardware and Software Requirements

The ATM SUBNET requires that the system be configured with the ATM ServerNet Access Platform software and hardware.

The TCP/IP SCF product module version G06.00 or later is required to add and manage SUBNETs of type ATM.

To use ATM on the Integrity NonStop NS-series server, you must connect to an IOMF2 CRU in a NonStop S-series I/O enclosure. For more information, see the *Introduction to Networking for HP Integrity NS-Series Servers*.

Viewing ARP and ATMARP Tables

The ARP and ATMARP tables can be viewed and changed using SCF. The ENTRY object allows you to view the ATMARP tables. ARP and ATMARP table entries added using SCF are named (such as #arp1); whereas, table entries added dynamically by the TCP/IP process are not named. In cases of unnamed table entries, you can enter the IP address of the entry to view or delete the individual entry. For more information about the SCF commands for viewing and altering the ARP and ATMARP tables, see Section 4, SCF Reference.

3 Configuring the NonStop TCP/IP Subsystem

This section discusses a set of command files that you can use to start and configure your NonStop TCP/IP environment over a network connected to a NonStop system. This section also provides information about networking services provided by NonStop TCP/IP and explains how to configure the OSS environment to use NonStop TCP/IP. See Appendix A, Configuration Reference, for reference information about the DNS server and the standard resource record format.

LAN-Based Connection

Configuration 1: Startup Files for a Host in a Basic NonStop TCP/IP Environment on page 3-2 and Configuration 2: Startup Files for a Host in a Subnet Addressing Environment on page 3-8 show the startup command files for two, complex, NonStop TCP/IP, LAN-based environments. You can perform the startup procedure manually, command by command, but you will find it much more efficient to build a set of command files that you can run whenever you want to start your NonStop TCP/IP environment. For a given system, you typically use two files for starting and configuring your NonStop TCP/IP environment:

- TCPIPUP2 is a TACL command file that serves as the main command file. TCIPUP2 calls the other files and brings up the NonStop TCP/IP subsystem by:
  - Accessing several SCF files that add, configure, and start NonStop TCP/IP objects
  - Starting the NonStop TCP/IP, LISTNER, and TELSERV processes

  TCPIPUP2 also sets the HOSTS file parameter if you do not have a domain name server
- SCFSBNT adds, configures, and starts the SUBNETs and routes.

These are the basic files that are used for configuring the NonStop TCP/IP software. You may have more files than this depending upon the size of your NonStop TCP/IP environment.

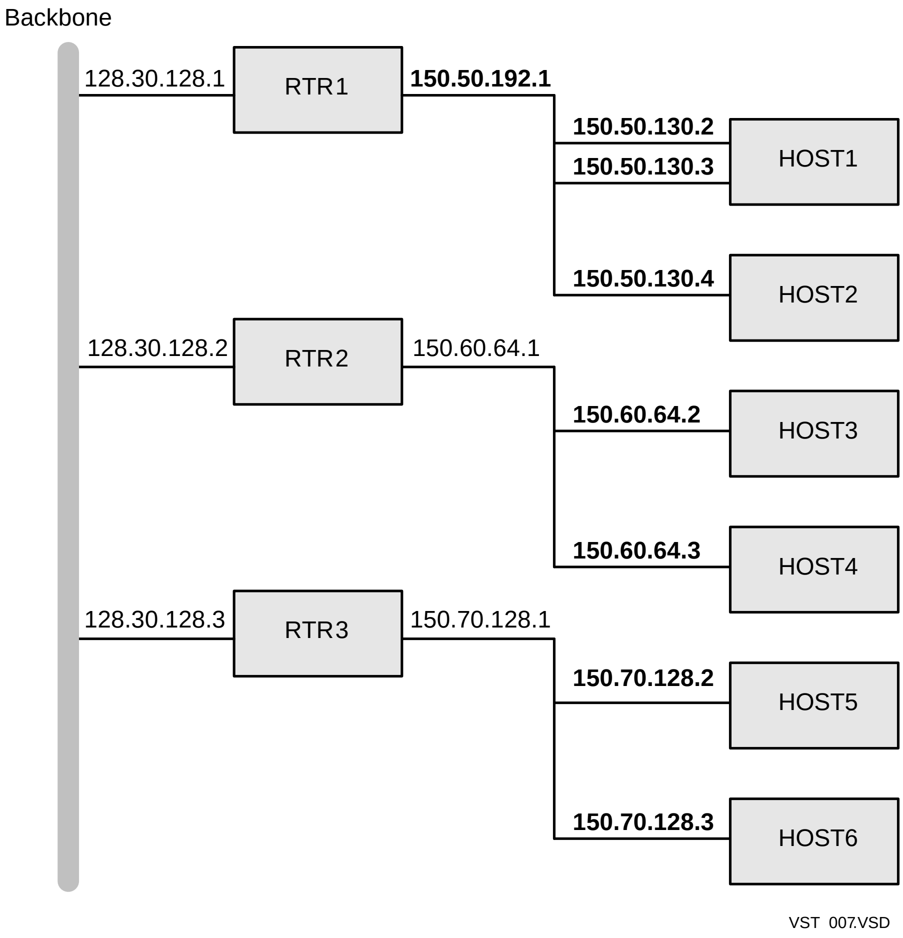
Note. You must understand the basics of SCF to optimize your understanding of the configuration of a NonStop TCP/IP environment. If you are not already familiar with the basics of using SCF, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Configuration 1: Startup Files for a Host in a Basic NonStop TCP/IP Environment

The sample environment shown in Figure 3-1 consists of a backbone communications link with three industry-standard routers connected to three different networks. These networks are made up of other NonStop systems. Because IP addresses are Class B, network-routing decisions are based upon the first two octets of each IP address.

The sample configuration provides the startup files for the NonStop system HOST1. The specific IP addresses entered in these files appear in boldface. For router configuration, consult the documentation provided by your vendor.

Figure 3-1. Basic NonStop TCP/IP Environment



HOST1 is a NonStop system that has two Gigabit Ethernet 4-port ServerNet adapters (G4SAs). (For more information about configuring G4SAs, see the *LAN Configuration and Management Manual* and the *Gigabit Ethernet 4-Port Adapter Installation and Support Guide*.) The two G4SAs attached to the same network appear as two separate hosts to the rest of the network. Although they could be configured using the same NonStop TCP/IP process ($ZTC0), the sample configuration shows them using separate NonStop TCP/IP processes ($ZTC0 and $ZTC1).

HOST1 does not use a Domain Name server for resolving the names of other hosts into their corresponding IP addresses. Thus, a HOSTS file is part of this sample configuration.

The TCPIPUP2 File

The following TACL command file starts the processes, adds and starts subsystem objects through SCF, and sets appropriate parameters. To add comments, use the word “comment” or a double equal sign (==). Lines which call other files are discussed separately below.

Note. The default NonStop TCP/IP process is $ZTC0 so you do not need to specify defines and params for NonStop TCP/IP applications such as LISTNER and TELSERV for the first part of Example 3-1.

Note. The NonStop TCP/IP primary and backup processes must be configured in CPUs that have access to the SAC. See Hint on page 1-7.

Example 3-1. TCPIPUP2 for HOST1

```
comment  ==== TCPIPUP2 =========TCPIPUP2 ========
comment  TACL command file to bring up NonStop TCP/IP
comment  subsystem
comment  Use HOSTS file for name resolution; not DNS
         ADD DEFINE =TCPIP^HOST^FILE, FILE &
         $SYSTEM.ZTCPIP.HOSTS
comment  Initialize the NonStop TCP/IP processes
comment  (corresponds to HOST1 in Figure 3-1)
         TCPIP/NAME $ZTC0, NOWAIT, CPU 1/0
         TCPIP/NAME $ZTC1, NOWAIT, CPU 1/0
comment  ADD and START SUBNET
         SCF/IN $SYSTEM.TCPIP.SCFSBNT/
comment  Init LISTNER for FTPSERV, ECHOSERV, FINGSERV for $ZTC0
         LISTNER/NAME $LSN0,CPU 4,NOWAIT,IN $ZHOME, &
         OUT $ZHOME, TERM $ZHOME, PRI 160/3 &
         $SYSTEM.ZTCPIP.PORTCONF
comment  Initialize TELSERV process for $ZTC0
         TELSERV/NAME $ZTN0,CPU 2,NOWAIT,PRI 170/-BACKUPCPU 3
comment  Set TCPIP^PROCESS^NAME to $ZTC1 prior to invoking
comment  LISTNER and TELSERV
         ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC1
comment  Initialize listner for FTPSERV, ECHOSERV,
comment  and FINGSERV for $ZTC1
         LISTNER/NAME $LSN1,CPU 4, NOWAIT, IN $ZHOME, &
         OUT $ZHOME, TERM $ZHOME, PRI 160/5 &
         $SYSTEM.ZTCPIP.PORTCONF
comment  Initialize TELSERV process for $ZTC1
         PARAM TCPIP^PROCESS^NAME $ZTC1
         TELSERV/NAME $ZTN1,CPU 2,NOWAIT,PRI 170/-BACKUPCPU 3
comment  ====== END OF TCPIPUP2 ==== END OF TCPIPUP2 ==
```

Type	**Syntax**	**Description**
Line	`ADD DEFINE =TCPIP^HOST^FILE, FILE $SYSTEM.ZTCPIP.HOSTS`	Sets the =TCPIP^HOST^FILE define to point to the desired HOSTS file. When set, this define tells the DNR to use the HOSTS file to translate host names to IP addresses. For information about the RESOLVER, see RESCONF Details on page 3-37 and the *TCP/IP Programming Manual*.
Run Command	`TCPIP/NAME $ZTC0, NOWAIT, CPU 1/0`	Starts the TCP/IP process named $ZTC0 in processors 1 (primary) and 0 (backup).
Run Command	`TCPIP/NAME $ZTC1, NOWAIT, CPU 1/0`	Starts the TCP/IP process named $ZTC1 in processors 1 (primary) and 0 (backup). Note that TCP/IP starts at a priority of 200, regardless of the priority specified in the RUN command.
Line	`ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC1`	Sets the =TCPIP^PROCESS^NAME parameter to $ZTC1. If you set the =TCPIP^PROCESS^NAME parameter, you must do so before starting the LISTNER and TELSERV processes. By doing so, you allow a newly started LISTNER or TELSERV process to examine the parameter specification and know which TCP/IP process to use. $ZTC0 is the recommended first name to use for the NonStop TCP/IP process. $ZTC0 is also the default value for applications on a NonStop system. Note that no ADD DEFINE statement exists for $ZTC0 because the LISTNER and TELSERV processes use it automatically if there is no TCP/IP process name defined. Successive NonStop TCP/IP process names should be $ZTC1, $ZTC2, and so on.

Run Command	`LISTNER/NAME $LSN1,CPU 4, NOWAIT, IN $ZHOME, &`
	`OUT $ZHOME, TERM $ZHOME,PRI 160/3 $SYSTEM.TCPIP.PORTCONF`
		Starts the LISTNER process responsible for starting the ECHO, FINGER, and FTP servers when a client request is received by the LISTNER process. Consequently, you should run this process at a higher priority. This command also specifies the location of the PORTCONF file used to designate which ports this process is to listen to. This process requires privileged access to some NonStop TCP/IP ports: therefore, always start with a super group ID.

Type	Syntax	Description

Line	`TELSERV/NAME $ZTN0,CPU 2,NOWAIT,IN $ZHOME, &`
	`  OUT $ZHOME, TERM $ZHOME, PRI 170/-BACKUPCPU 3`
		Starts the Telserv process. When an application (such as Outside View) connects to the IP address of the TCP/IP process, Telserv presents a window with a Welcome Banner and Service Menu. You can start TACL using the Service Menu.

The SCFSBNT File

A SCFSBNT file adds and starts SUBNETs and routes. Example 3-2, this file begins by assuming the NonStop TCP/IP processes $ZTC0 and, later, $ZTC1. Consequently, these two NonStop TCP/IP processes must be started before the file is run. The specifications for Logical Interface objects, which provide an interface between TCP/IP and the LAN adapter, appear in boldface.

Example 3-2. SCFSBNT File for TCPIPUP2

```
=== SCFSBNT ===== SCFSBNT ==== SCFSBNT ========
== SCF command file to ADD and START SUBNETs
== This file is created to support HOST1 (Refer to
== Figure 3-1)
   ALLOW ALL ERRORS
   ALLOW ALL WARNINGS
== ADD AND START SUBNET $ZTC0.#SN0
   ASSUME PROCESS $ZTC0
   ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN01,IPADDRESS 150.50.130.2
== ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
== ADD ROUTES
   ADD ROUTE #ROU0, DESTINATION 128.30.0.0, GATEWAY 150.50.192.1
   ADD ROUTE #ROU1, DESTINATION 150.60.0.0, GATEWAY 150.50.192.1
   ADD ROUTE #ROU2, DESTINATION 150.70.0.0, GATEWAY 150.50.192.1
== START SUBNETS & ROUTES
   START SUBNET *
   START ROUTE *

==
== ADD AND START SUBNET $ZTC1.#SN1
==
   ASSUME PROCESS $ZTC1
== ADD SUBNET
   ADD SUBNET #SN1,TYPE ETHERNET,DEVICENAME LAN02,IPADDRESS 150.50.130.3
== HELP TCPIP ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
== HELP TCPIP START SUBNET
   START SUBNET *
== HELP TCPIP ADD ROUTE
== ADD ROUTES
   ADD ROUTE #ROU0, DESTINATION 0.0.0.0, GATEWAY 150.50.192.1
== START ROUTE
   START ROUTE *
==
=========== END OF SCFSBNT ============= END OF SCFSBNT ===========
```

For the ADD SUBNET command, you can assign the SUBNET name to be anything you like, provided the name is no more than seven alphanumeric characters long and begins with an alphabetic character. The DEVICENAME attribute, which specifies the logical interface (LIF) name associated with the adapter that the NonStop TCP/IP process will access, is required. (See Hint on page 1-7 for determining an appropriate LIF.) You can have more than one ServerNet adapter supporting the same NonStop TCP/IP process. You must have a unique IP address for each ServerNet adapter that is physically attached to your network. The IP address links the I/O process name and the NonStop TCP/IP process.

LOOPBACK

When the NonStop TCP/IP process is started, a SUBNET named #LOOP0 is added automatically. This SUBNET exists to provide loopback capability without requiring the use of the TCP/IP network. When this #LOOP0 SUBNET is created, it has an address of 0.0.0.0 in dotted decimal form. For correct operation, the address needs to be changed by using the command:

```
ALTER SUBNET #LOOP0, IPADDRESS 127.1
```

The address 127.1 or 127.0.0.1 is the standard for loopback operation.

ROUTE Objects

The routes are then added. A ROUTE object is added for each remote subnet destination with which this host will need to communicate. The ROUTE object specifies the destination network IP address and the router IP address to which this host is physically connected. You specify the router address in the GATEWAY attribute of the ADD ROUTE command.

GATEWAY Attribute

In this sample environment, there are potentially three destination networks to which HOST1 could communicate, and the router address specified in the GATEWAY attribute is the same for each route. As you can see from reviewing Figure 3-1, HOST1 must route a datagram destined for one of those three networks through RTR1, which has the IP address of 150.50.192.1. This method is especially useful when you have multiple routers to multiple networks. When all the routing is through a single router, however, there is a simpler way to set up your routing.

Default routing establishes a single route as the default route. This action is particularly useful when you know that most of your TCP/IP traffic is going through a single router, as in the case shown in Example 3-2. The single route added for the second NonStop TCP/IP process ($ZTC1) in Figure 3-1 implements default routing. What indicates that this is a default route is the use of 0.0.0.0 to designate the destination network IP address. You can add more routes for networks that cannot be reached by using the default route.

Subnet Mask

The subnet mask is not altered in this file because the default mask of %hFFFF0000 is adequate for a class B IP address on a network without subnets. The first two octets of the IP address are adequate for determining the proper network.

Be aware that adding SUBNETs and routes is not the same as implementing subnetting. This sample configuration configures a host (HOST1) connected to a single network which is connected to another network through a router (RTR1). Neither of these two networks has implemented subnetting. (Subnetting is shown and explained in Appendix B, NonStop TCP/IP Processes and Protocols.) The use of the SUBNET object simply means implementing NonStop TCP/IP within the environment.

Next the SUBNETs and routes of the NonStop TCP/IP processes $ZTC0 and $ZTC1 are started.

The HOSTS File

The HOSTS file is used in the absence of a Domain Name Server for resolving the common names of hosts into their corresponding IP addresses. (The HOSTS file shown is customized for the purpose of this sample environment.)

Example 3-3. HOSTS File for TCPIPUP2

```
########## HOSTS FOR HOST1 ########## HOSTS FOR HOST1 ############
# Filename = \CB1.$SYSTEM.ZTCPIP.HOSTS
# Date     = January 31/93
127.0.0.1     me loop
150.50.130.2  lan01 con1
150.50.130.3  lan02 con2
150.50.130.4  host2 corp2
150.60.64.2   host3 corp3
150.60.64.3   host4 corp4
150.70.128.2  host5 RD1
150.70.128.3  host6 RD2
###########END OF HOSTS ##################END OF HOSTS ###########
```

All text following a pound sign (#) is comment text. Use comment text to note revisions made to the file, the name of the I/O process, the hardware address of the ServerNet adapter, and so on. Knowing the hardware address of the ServerNet adapter helps you when you test the network through the Services Manager (TSM).

Begin the IP addresses of the hosts in column one of the HOSTS file. Separate the host name from the address by at least one space. You may have as many aliases as can fit on a single entry line.

The lines in the HOSTS file:

```
127.0.0.1    me  loop
150.50.130.2 LAN01 lan01 con1
150.50.130.3 LAN02 lan02 con2
```

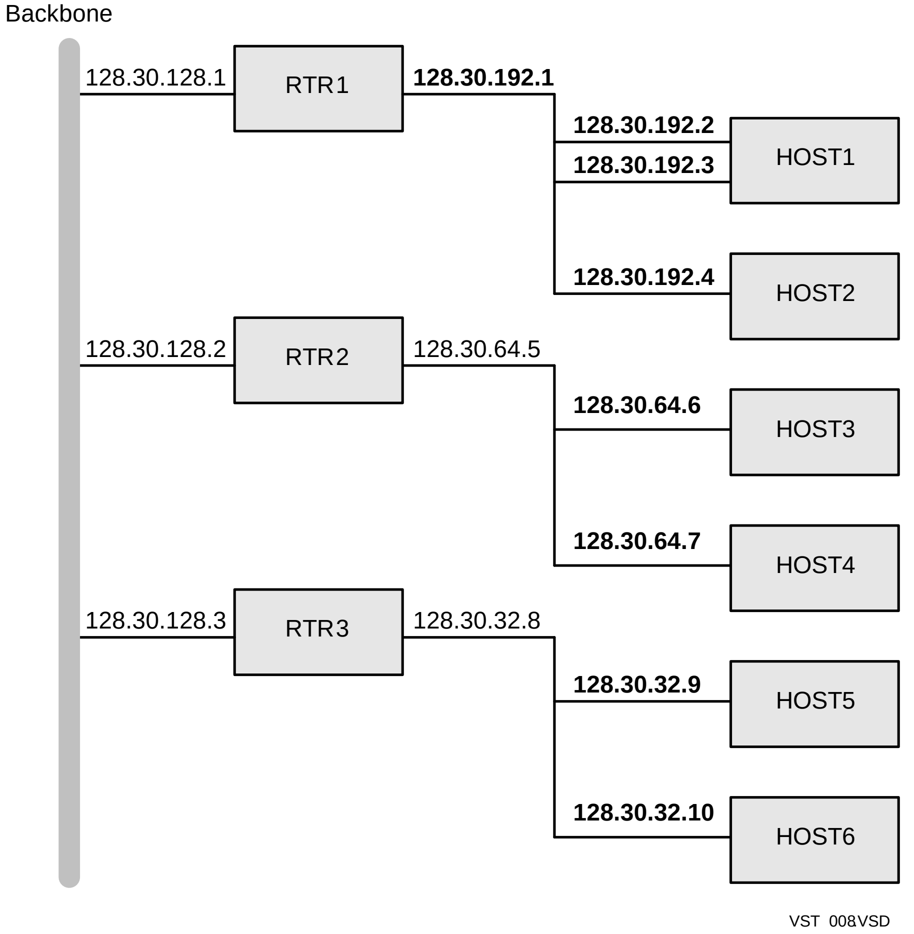
provide flexibility in testing the environment. When you use the ECHO service to send an echo datagram to me or loop, you are testing the client and server capabilities of your own ECHO service. If you send an ECHO datagram to lan01 or lan02, you also test the actual physical network connection for your HOST1 NonStop TCP/IP environment.

Configuration 2: Startup Files for a Host in a Subnet Addressing Environment

This environment consists of a backbone communications link that has three routers leading to three subnets. Since subnet addressing is being used, the subnet mask must be altered on those networks from the default of %hFFFF0000 to%hFFFFFF00. This means that routing decisions must be based on the first three octets of the IP address rather than on the first two. However, subnetting will not be implemented on the backbone so that you can see how subnetting can coexist with networks that are not using subnetting.

Only those configuration files which must be changed from the first sample configuration will be shown; and only those lines within the files that are different will be discussed. For a complete listing of the files and line-by-line explanation, see the configuration files for HOST1 (TCPIPUP2 for HOST1 on page 3-3.) For router configuration, consult the documentation provided by your vendor.

Figure 3-2. NonStop TCP/IP Environment Using Subnet Addressing



HOST1 is a NonStop system that has two G4SA adapters which appear as two separate hosts to the rest of the network. (You also can alias a subnet to use multiple IP addresses for applications such as web-servers. For more information about alias IP addresses, see the ALTER SUBNET Command on page 4-31.) Although you could use the same TCP/IP process ($ZTC0) to configure the hosts, this sample configuration uses separate NonStop TCP/IP processes ($ZTC0 and $ZTC1).

Since the actual physical connections are not changing in this sample configuration, the TCPIPUP2 file shown in Example 3-1 for HOST1 can be used here. That file is therefore not shown in this subsection. The first file that has significant changes, and that therefore must be shown, is the SCFSBNT file.

The SCFSBNT File

Use the SCFSBNT file shown in Example 3-4 to add and start SUBNETs and routes. As stated earlier, SUBNETs and routes are subordinate to the NonStop TCP/IP process. Therefore, SCFSBNT begins by assuming the NonStop TCP/IP process $ZTC0. The NonStop TCP/IP process must be started to add SUBNETs and routes.

The specifications for Logical Interface objects, which provide an interface between TCP/IP and the E4SA adapter, appear in boldface.

Example 3-4. Second SCFSBNT File for TCPIPUP2 (Subnetting)

```
=== SCFSBNT FOR HOST1======== SCFSBNT FOR HOST1========
== SCF command file to ADD and START SUBNETs
== This file is created to support HOST1 (Refer to Example 3-1)
   ALLOW ALL ERRORS
   ALLOW ALL WARNINGS
==
== ADD AND START SUBNET $ZTC0.#SN0
==
   ASSUME PROCESS $ZTC0
== HELP TCPIP ADD SUBNET
   ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN01,IPADDRESS 128.30.192.2
== HELP TCPIP ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
   ALTER SUBNET #SN0, SUBNETMASK %HFFFFFF00
== HELP TCPIP ADD ROUTE
== ADD ROUTES
   ADD ROUTE #ROU0, DESTINATION 128.30.0.0, GATEWAY 128.30.192.1
== HELP TCPIP START SUBNETS & ROUTES
   START SUBNET *
   START ROUTE *
==
== ADD AND START SUBNET $ZTC1.#SN1
==
   ASSUME PROCESS $ZTC1
== HELP TCPIP ADD SUBNET
   ADD SUBNET #SN1,TYPE ETHERNET,DEVICENAME LAN02,IPADDRESS 128.30.192.3
== HELP TCPIP ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
   ALTER SUBNET #SN1, SUBNETMASK %HFFFFFF00
== HELP TCPIP START SUBNET
   START SUBNET *
== HELP TCPIP ADD ROUTE
== ADD ROUTES
   ADD ROUTE #ROU0, DESTINATION 0.0.0.0, GATEWAY 128.30.192.1
== HELP TCPIP START ROUTE
   START ROUTE *
==
=========== END OF SCFSBNT ============= END OF SCFSBNT ===========
```

This SCFSBNT file differs from that in Example 3-5. The most obvious change is that the IP address in the ADD SUBNET command now matches the configuration in Figure 3-2. Also, two additional ALTER SUBNET commands in this file alter the SUBNET mask for each SUBNET added.

The default subnet mask is %HFFFF0000, so you must distinguish the subnet number from the host number. Since this sample configuration uses Class B addressing, two octets are left over for the host address. On a Class B network, you usually split the subnet number from the host number on the third octet boundary. Thus, you have three octets for the network address and one octet for the host address.

The configuration in SCFBNT allows HOST1 to communicate with the three subnets shown in Figure 3-2:

```
128.30.192.0 (the local subnet for HOST1 and HOST2)
128.30.64.0  (a remote subnet for HOST3 and HOST4)
128.30.32.0   (a remote subnet for HOST5 and HOST6)
```

When you add SUBNET objects to the NonStop TCP/IP process, the process needs to know what mask is being applied to messages on that SUBNET. Use the ALTER SUBNET commands to change the subnet mask from the default to %HFFFFFF00.

The routes also reflect the new configuration. The ADD ROUTE command for $ZTC0 provides access to the backbone and subnets 128.30.32.0 and 128.30.64.0. The routing for the second NonStop TCP/IP process ($ZTC1) continues to use a default address.

The HOSTS File

The HOSTS file, shown in Example 3-5, contains address changes for our second sample configuration. You can use this HOSTS file for all the hosts in the sample configuration. Communication destined for HOST1 defaults to the line $LAN01, but you can call HOST1B to address the other line.

Example 3-5. Second HOSTS File for TCPIPUP2

```
########## HOSTS FOR HOST1 ########## HOSTS FOR HOST1 ############
# Filename = \CB1.$SYSTEM.ZTCPIP.HOSTS
127.0.0.1     me loop
128.30.192.2  host1 h1
128.30.192.3  host1b h1b
128.30.192.4  host2 corp2
128.30.64.6   host3 corp3
128.30.64.7   host4 corp4
128.30.32.9   host5 RD1
128.30.32.10  host6 RD2
###########END OF HOSTS ##################END OF HOSTS ###########
```

Further Subnetting Options

Once again, assume that you have a Class B IP network address. By reasonably projecting growth, you determine that you can split the subnet and host IDs on the octet boundary to provide for 254 subnets with 254 possible hosts on each subnet. The subnet mask would be:

```
11111111  11111111  11111111  00000000
```

that is, %HFFFFFF00.

Over the next year, however, your organization decides to expand its operations and the number of its networks. You no longer need as many hosts attached to each subnet as you originally planned on; however, you do need to fit more than 254 subnets into your addressing scheme.

This plan will be simple to implement if you assign the subnet addresses beginning with the high-order bits and host IDs beginning with the low-order bits. In such a case, the first few subnet addresses would be 128, 64, 192, and so on. In this case, you might need to change the subnet number and the subnet mask.

In this sample environment, suppose that you need to combine subnets 128.30.192.0 and 128.30.64.0 due to an administrative consolidation. If you assigned the subnet addresses and host addresses as recommended, you may have enough room for the added hosts by simply adjusting your subnet mask.

For example, the current subnet mask looks like this in binary form:

```
11111111  11111111  11111111  00000000
```

For a Class B address, the first two octets determine the network address. In the subnet mask above, you use the third octet to specify the subnet address. These subnets are 192, 64, and 32. The corresponding host addresses are numbered consecutively throughout the network when possible. You are numbering the subnets using the high order bits first and numbering the hosts using the low order bits first. So, the binary representations of two host addresses, for example, HOST1 and HOST3, respectively are:

```
10000000  00011110  11000000  00000010
10000000  00011110  01000000  00000110
```

Their addresses in dotted decimal form are:

```
128.30.192.2
128.30.64.6
```

In the third octet, you use the high-order bits first (128, 192, and 64); in the fourth octet for the host ID, you use the low-order bits first (1, 2, 3, and so on). By using this subnet and host addressing method, when you need to increase the number of subnets or hosts beyond the octet boundary, you have room to do so.

However, if you started numbering the subnets with the low order bits first (1, 2, 3, and so on), then needed to place more that 254 hosts on that subnet, you would have to reconfigure each host on the subnet.

If you leave room for growth in the number of subnets and hosts, you substantially reduce the number of changes and reconfiguration needed on each host.

WAN-Based Connections

This subsection describes startup files for configuring X.25 connections.

Note. As for LAN-based connections, the SLSA subsystem must be operational before you can configure SUBNETs of type X25.

Configuring an X25AM I/O Process on NonStop S-Series Systems and Integrity NonStop NS-series Systems

If your application requires you to configure a TCP/IP SUBNET of type X25, you must have the X25AM I/O process configured first. See the *X25AM Configuration and Management Manual* for information about configuring the X25AM I/O process but do not be confused by the fact that the X25AM subsystem requires you first to start the NonStop TCP/IP subsystem. On NonStop S-series systems and Integrity NonStop NS-series systems, NonStop TCP/IP provides connectivity to the SWAN concentrator for all I/O processes.

If you have followed the steps to bring up your NonStop S-series system or Integrity NonStop NS-series system, NonStop TCP/IP is started.

Note. If you are using X25 from an Integrity NonStop server, the system must be connected to an IOMF2 customer replaceable unit (CRU) in a NonStop S-series system I/O enclosure to access the hardware. (The software for X25 is available on the Integrity NonStop NS-series server.) See the *Introduction to Networking for HP Integrity NonStop NS-Series Servers* for more information about interconnectivity between the Integrity NonStop NS-series system and NonStop S-series system for X25 communications.

Start the X25AM subsystem, then add TCP/IP SUBNETs of type X25.

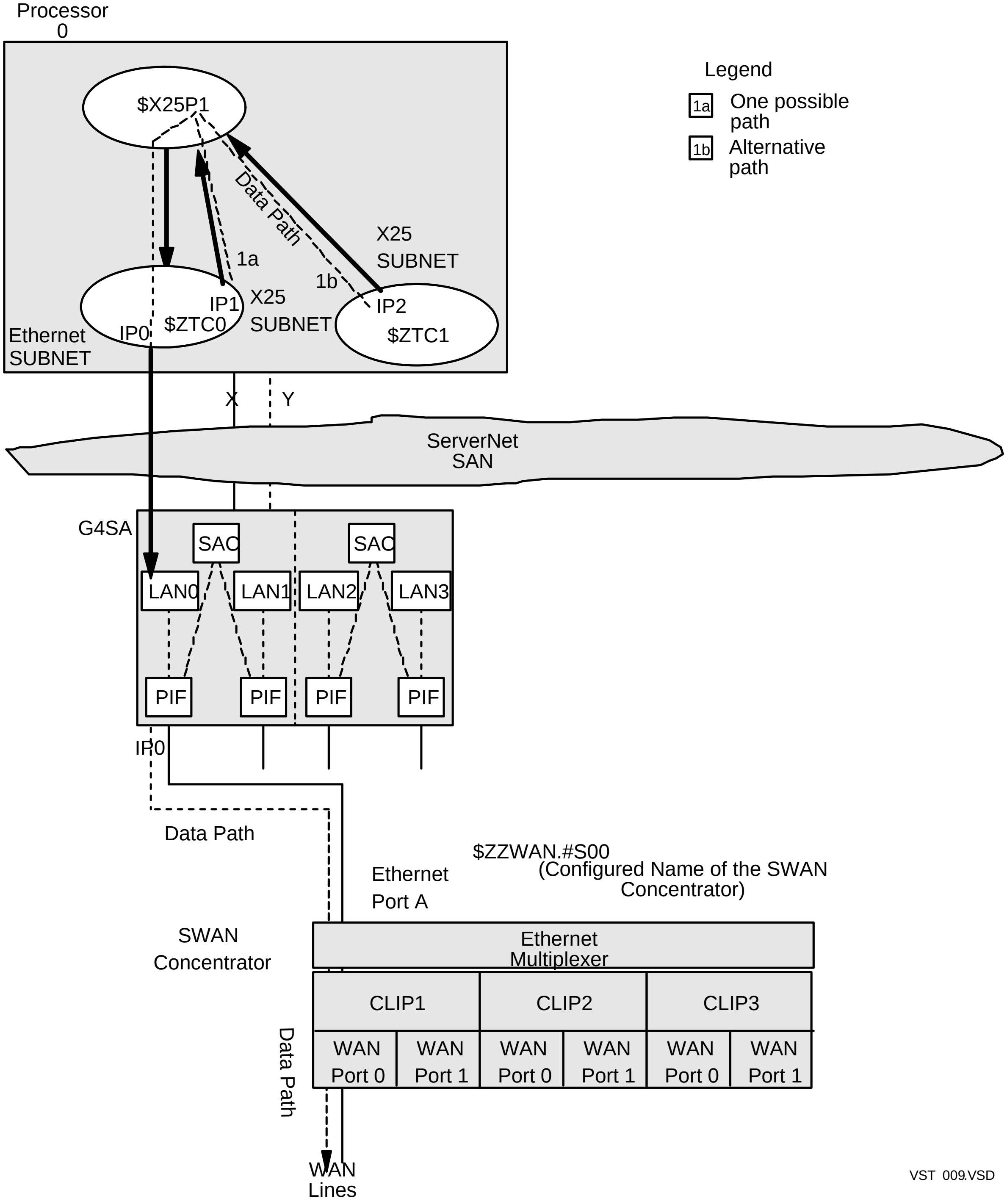
If you are re-configuring your system after a cold-load,

1. Start your NonStop TCP/IP subsystem using the methods described in Section 1, Configuration Quick Start and in Example 3-1 through Example 3-16 of this section
2. Start an X25AM I/O process by following the procedures described in the *X25AM Configuration and Management Manual*.
3. Return to this manual and proceed with Configuration 3: Startup Files for Two Hosts Using an X25AM-Based Connection on page 3-15.

Figure 3-3, Configuring a Subnet of Type X25, shows two TCP/IP processes ($ZTC0 and $ZTC1) using X25 SUBNETs (IP1 and IP2). The requirement of providing a TCP/IP interface to the LAN for X25AM is shown with IP0 which is an Ethernet SUBNET running in $ZTC0.

X25AM accesses the X.25 line through SUBNET IP0, and SUBNETs IP1 and IP2 access the X.25 line through $X25P1 (the X25AM I/O process).

Figure 3-3. Configuring a Subnet of Type X25



Configuration 3: Startup Files for Two Hosts Using an X25AM-Based Connection

This subsection describes how to configure NonStop TCP/IP hosts that connect through an X.25-based network.

Figure 3-4 illustrates an X.25 public data network (PDN) that connects a local NonStop TCP/IP host (\LA) to a remote NonStop TCP/IP host (\NY) across the country in New York. To make this connection possible, each host is given both an IP address and an X.121 source address.

\LA has the following two addresses:

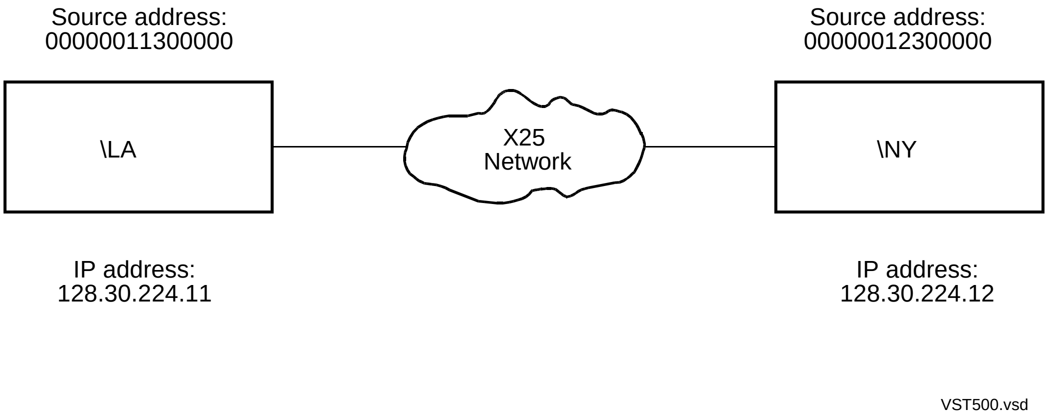
- IP address: 128.30.224.11
- X.121 source address: 00000011300000

\NY has the following two addresses

- IP address: 128.30.224.12
- X.121 source address: 00000012300000

Sample startup files are provided for each side of the connection shown in the figure. These files show both NonStop TCP/IP and X25AM configuration steps, including the step that each host uses to map the IP and X.121 addresses for its destination. Note that the sample configuration does not show the HOSTS files for \LA and \NY. For example HOSTS files, see The HOSTS File on page 3-7 in Configuration 1: Startup Files for a Host in a Basic NonStop TCP/IP Environment and The HOSTS File on page 3-11 in Configuration 2: Startup Files for a Host in a Subnet Addressing Environment.

Figure 3-4. NonStop TCP/IP Hosts Connected Through an X.25 Network



Before proceeding, look closely at the details of \LA and \NY.

- Each host (\LA and \NY) physically connects to an X.25 network through a line on a SWAN concentrator. \LA1 uses line $X250 and \NY uses line $X25A.

- Each host has an X25AM I/O process to control its SWAN concentrator.
- Each I/O process to has six subdevices (SUs). A maximum of 128 SUs per line is possible. The X25AM I/O process uses these SUs to provide multiple simultaneous connections over the X.25 line. SCF will be used to configure and start these SUs.
- Each host has its own NonStop TCP/IP process: $ZTC1 on \LA uses $X250; $ZTC2 on \NY uses $X25A.
- $ZTC1 has a SUBNET to represent its connection over $X250 to the X.25 network. $ZTC2 has a SUBNET to represent its connection over $X25A to the X.25 network.
- Each NonStop TCP/IP process automatically adds a ROUTE object when its respective SUBNET is added.
- Each NonStop TCP/IP process provides the platform upon which applications can establish and use connections. These connections are established through the use of ports (either well-known or application-specific). Applications make simultaneous use of these ports by using the sockets interface.

To configure each host with its connection to an X.25 network, do the following:

1. Configure the X.25 line with a compatible set of attributes.
2. Configure the X.25 line with a set of SUs. Each SU must have an appropriate set of attributes.
3. Start the X.25 line.
4. Start a NonStop TCP/IP process, and configure it with a SUBNET object and appropriate ROUTE objects.

These tasks and their details are new, but the basic configuration tasks are the same as those for a LAN-based environment:

- Configure the I/O process and its line(s). For X25AM, you configure line attributes and SUs.
- Start a NonStop TCP/IP process and configure SUBNET objects and ROUTE objects. For X.25-based SUBNETs, you specify a SUBNET type of X.25.

Note. Other SUBNET types exist to support subsystems other than X25AM. For example, LAN-based SUBNET objects require a SUBNET type of ETHERNET or SNAP. ATM-based SUBNET objects require a SUBNET type of ATM.

- Start a LISTNER process for each NonStop TCP/IP process.

Now, examine the set of command files that will configure the X.25 network. The first set of command files configures \LA; the second set configures \NY.

Configuring an X.25 Connection for \LA

Consider the organization of the set of command files that you will be examining. There are four files to look at:

- UP1 is the main command file. It invokes X25SUBU1, TCPUP1, and STATUS1.
- X25SUBU1 configures the X.25 line and SUs, then starts the line.
- TCPUP1 sets the appropriate ADD DEFINEs and starts the TCP/IP process. It then invokes SCF against TCPSUBU1 which configures SUBNET objects and ROUTE objects. Then TCPUP1 starts the LISTNER process. Next, it invokes SCFWNDW1 which configures a TELSERV WINDOW object. Then, TCPUP1 invokes a TACL process against the WINDOW. The TCPUP1 counterpart from the LAN-based configurations is TCPIPUP.
- STATUS1 displays status information about the X.25 line and its SUs and about the TCP/IP process and its SUBNETs and routes.

UP1: The Main Command File

The command file UP1 acts as the root command file and invokes all the second-level subordinate command files. Example 3-6 shows a listing of the command file for UP1.

Example 3-6. Main Up File for X.25

```
== TACL command file to bring up TCP/IP over X25 for \LA.
== Set default volume and subvolume
   Volume $CB21.TCPIPX25
== Add and start PDN X25 subdevices
   scf/in X25SUBU1/
   delay 10 sec
== Start TCP/IP Subsystems.
   OBEY TCPUP1
== Status everything
   SCF/in STATUS1/
```

Command	Description
`VOLUME $CB21.TCPIPX25`	sets the default volume and subvolume to point to the subvolume that contains the command files.
`SCF /IN X25SUBU1/`	invokes SCF against the command file X25SUBU1. This command file configures X.25 line attributes and SUs.

Command	Description
`OBEY TCPUP1`	invokes the command file to start and configure TCPIP, LISTNER, and TELSERV.
`TCISCF /IN STATUS1/`	invokes SCF against the command file STATUS1 to display status information.
`SCP /NAME $ZNET, NOWAIT, CPU 1/0; AUTOSTOP -1`	starts SCP and keeps it running.

X25SUBU1

The command file X25SUBU1 adds and starts X.25 subdevices. Example 3-7 shows a listing of the command file for X25SUBU1.

Example 3-7. Command File for Adding X.25 Subdevices

```
== SCF command file to add and start X25 subdevices for TCPIP on
== \LA.
   allow all
== SWAN concentrator: Lines 0($X250) and 1($X251) available.
   assume line $x250
   abort, sub all

   alter, netid x25xa, srcaddr "00000011300000", &
          Framemode DTE, callingaddr on, SVCRANGE (1,128)

   add su #tcpip1,devtype (9,0),protocol ptp,recsize 1024
   add su #tcpip2,like #tcpip1
   add su #tcpip3,like #tcpip1
   add su #tcpip4,like #tcpip1
   add su #tcpip5,like #tcpip1
   add su #tcpip6,like #tcpip1

   start line, sub all
```

SCF is invoked to run the commands contained in the command file X25SUBU1. This file contains the commands needed to alter line attributes and add SUs.

Command **Description**

`ALLOW ALL`

instructs SCF to permit all errors to occur without exiting the command file.

`ASSUME LINE $X250`

sets the assumed object type to LINE (specifically $X250 is the device name of the X.25 line connected to the X.25 network).

By setting the assumed object to $X250, all subsequent commands apply to this object, unless explicitly specified otherwise.

The command sets attributes for this line's connection to a PDN:

```
ALTER , NETID X25xa, SRCADDR
"00000011300000", &
  FRAMEMODE DTE, CALLINGADDR ON,
SVCRANGE (1, 128)
```

You must set most attributes to values that depend on the type of the network and the type of services you have contracted for. Refer to the *X25AM Configuration and Management Manual* for a discussion of the most commonly used attributes.

SRCADDR specifies the source address of the line. Within an X.25 network, an X25AM I/O process acts as Data Terminating Equipment (DTE), while the network service provider acts as Data Circuit Equipment (DCE). Thus, in this sample configuration, FRAMEMODE is set to DTE in the X25AM configuration files of each of the NonStop hosts (see X25SUBU2 in Configuration of an X.25 Connection for \NY).

Note. In laboratory test situations, a real X.25 network might not exist to provide DCEs. If \LA and \NY were connected in this way, one of them would need to function as a DCE. Thus, you would specify FRAMEMODE DCE for either \LA or \NY.

The following command sets attributes for this line's connection to an X.25 network:

```
ADD SU #TCPIP1, devtype (9,0), protocol ptp, recsize 1024
```

The NonStop TCP/IP process requires the attribute devtype (9,0), protocol ptp, and recsize 1024.

Note. You should not configure the X.25 SU attribute DESTADDR. The NonStop TCP/IP process programmatically sets the attribute DESTADDR prior to requesting that an X.25 call be made. You can, however, specify ADDRMAP in your NonStop TCP/IP configuration file (see TCPSUBU1). ADDRMAP contains an entry for the remote IP address and the remote source address.

The following commands add five more SUs to this line. These SUs allow the NonStop TCP/IP process to communicate with the X25AM I/O process to establish an X.25 circuit over which the TCP/IP communication can occur:

```
ADD SU #TCPIP2, like #TCPIP1
ADD SU #TCPIP3, like #TCPIP1
ADD SU #TCPIP4, like #TCPIP1
ADD SU #TCPIP5, like #TCPIP1
ADD SU #TCPIP6, like #TCPIP1
```

A TCP/IP process uses up to 32 SUs on each X25AM line; so, you can create up to 32 SUs. The naming convention of these SUs is assumed by the TCP/IP process to be #TCPIP*n*, where *n* is a number from 1 through 32.

When you configure a NonStop TCP/IP process, you add a SUBNET object type of X.25. An attribute of the X.25 SUBNET is the X.25 line name ($X250 in this case). When that SUBNET object is started, the NonStop TCP/IP process communicates with the X25AM I/O process by trying to open SUs that have the name #TCPIP*n*. The NonStop TCP/IP process continues opening SUs until NonStop TCP/IP opens the last SU (#TCPIP32).

TCPUP1

The command file TCPUP1 starts and configures the TCPIP, LISTNER, and TELSERV processes. Example 3-8 shows a listing of the command file for TCPUP1.

Example 3-8. Service Processes Up File for X.25

```
== TACL command file to bring up TCP/IP subsystem on \LA
== Use HOSTS file for name resolution instead of DNS
        ADD DEFINE =TCPIP^HOST^FILE, FILE $CB21.TCPIPX25.MJHOSTS
== Initialize first TCP/IP process
        TCPIP /NAME $ZTC1, NOWAIT, CPU 1/2
== Prior to starting LISTNER and TELSERV, use ADD DEFINE
statement
== to specify TCPIP process name
        ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC1
== Initialize TCPIP SUBNETs for X25
        SCF /IN TCPSUBU1/
== Initialize LISTNER process for FTPSERV, ECHOSERV, FINGSERV
        LISTNER /NAME $ZPORT, NOWAIT, PRI 160, CPU 2/3&
        PORTCONF

== Initialize TELSERV process
        PARAM TCPIP^PROCESS^NAME $ZTN1
        TELSERV/NAME $ZTN1,CPU 2,NOWAIT,PRI 170/-BACKUPCPU 3
```

Command	Description
`ADD DEFINE =TCPIP^HOST^FILE, FILE MJHOSTS`	sets the ADD DEFINE =TCPIP^HOST^FILE. Whenever a TCP/IP process must convert from a host name to an IP address, the TCP/IP process either uses the DNS or a HOSTS file like MJHOSTS. When the ADD DEFINE =T=TCPIP^HOST^FILE is set, the TCP/IP process uses the file as indicated. This sample configuration does not provide sample HOSTS files. For sample HOSTS files, see The HOSTS File on page 3-11.
`TCPIP /NAME $ZTC1, NOWAIT, CPU 1/2`	starts a TCP/IP process (this particular command assigns the process name $ZTC1). CPU 1 causes the TCP/IP process to run as a NonStop process pair, with the primary process in CPU 1 and the backup process in CPU 2.
`ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC1`	sets the ADD DEFINE =TCPIP^PROCESS^NAME to the process name $ZTC1. Whenever a TCP/IP process is started using a process name other than the default ($ZTC0), you must set ADD DEFINE prior to running any TCP/IP application (for example, LISTNER or TELSERV). Each application reads the ADD DEFINE =TCPIP^PROCESS^NAME to determine the process name to communicate with. If this ADD DEFINE is not set, LISTNER and TELSERV processes try to communicate with $ZTC0. $ZTC0 may be the wrong process or it may not even exist.
`SCF /IN TCPSUBU1/`	invokes SCF to run the commands in the file TCPSUBU1; this file contains the commands to add the necessary SUBNET and ROUTE objects to $ZTC1.

Command	Description
`LISTNER /NAME $ZPORT, NOWAIT, PRI 160, CPU 2/3 PORTCONF`	invokes the LISTNER process using a process name of $ZPORT. $ZPORT uses $ZTC1 (TCP/IP process) and listens for connection requests from the processes listed in the file PORTCONF. CPU 2 starts the LISTNER in CPU 2. The LISTNER can run as a NonStop process pair; therefore, CPU 3 is specified as backup. The line: `TELSERV/NAME $ZTN1,CPU 2,NOWAIT,PRI 170/-BACKUPCPU 3` starts the Telserv process in CPU 2, with CPU 3 specified as backup.

Note. To add an X.25 network connection to an existing NonStop TCP/IP subsystem, you do not need the commands to start TCPIP, LISTNER, and TELSERV processes. But make sure that the SCF command file TCPSUBU1 assumes the correct NonStop TCP/IP process name.

TCPSUBU1

The command file TCPSUBU1 adds an X.25 SUBNET and ADDRMAP specification to the NonStop TCP/IP process $ZTC1. Example 3-9 shows a listing of the command file TCPSUBU1:

Example 3-9. Command File for Adding X.25 Subdevices

```
== SCF command file to add a SUBNET to a TCP/IP process for an
X25 == line
  allow all errors
== SWAN concentrator: line $X250: IP-Address 128.30.224.11
  assume process $ztc1
  alter subnet #loop0,ipaddress 127.1
  add subnet #pdn1,type x25,devicename $X250, &
     ipaddress 128.30.224.11
  alter subnet #pdn1, subnetmask %HFFFFFF00
  add addrmap #a1, ipaddress 128.30.224.12,   &
      x121addr "00000012300000"
  start subnet *
  start route *
```

Command	Description
`ALLOW ALL ERRORS`	permits SCF to continue running commands in the command file regardless of errors.
`ASSUME PROCESS $ZTC1`	sets an SCF assumed object. All subsequent commands apply to this object unless explicitly entered otherwise.
`ALTER SUBNET #LOOP0, IPADDRESS 127.1`	sets the IP address for the loopback SUBNET.
`ADD SUBNET #PDN1, TYPE X25, DEVICENAME $X250, IPADDRESS 128.30.224.11`	adds an X25 SUBNET object that has the name #PDN1 to the TCP/IP process $ZTC1.

Note. To obtain a list of running X25AM lines to use in the DEVICENAME field, issue the LISTDEV TYPE 61 at the SCF prompt. The X25AM lines are listed in the Names column.

Subnet names start with a pound sign (#) followed by an alpha character. The remaining six characters can be any combination of alphanumeric characters. In this case, (#PDN1) PDN denotes Public Data Network.

$X250 is the X.25 line to service connections.

128.30.224.11 is the IP address associated with the \LA connection to this X.25 PDN.

`ALTER SUBNET #PDN1, SUBNETMASK %HFFFFFF00`	sets the subnet mask so that the first three octets of an IP address determine the network address.

`ADD ADDRMAP #A1, IPADDRESS 128.30.224.12,   &`
`      X121ADDR "00000012300000"`
	maps the IP address of the remote host with its X.121 source address.

Note. In this sample configuration, the X.121 source addresses of \LA and \NY specified in the SCRADDR attribute were derived from their IP addresses in accordance with the specification contained in the Military Standards for X.25. Thus, NonStop TCP/IP could establish a connection without the ADDRMAP specification. However, if you were to specify source addresses that were not derived in this way, you would have to include an ADDRMAP attribute in TPSUBU1and TPSUBU2.

`START SUBNET *`	places all the $ZTC1 SUBNETs into service.
`START ROUTE *`	places all the $ZTC1 routes into service.

Note. When you add a SUBNET object, you automatically create a default ROUTE object. No explicit ADD ROUTE command is necessary. Thus, this sample TCPSUBU1 file contains no ADD ROUTE command. The TCPSUBU1 file does, however, contain the required START ROUTE command that starts the default routes. For more information, see ROUTE Object Type on page 4-4, ADD SUBNET Command on page 4-21, and ADD ROUTE Command on page 4-18.

STATUS1

The command file STATUS1 displays status information about the $X250 SUs, the NonStop TCP/IP process $ZTC1, and the $ZTC1 SUBNET and ROUTE objects. Example 3-10 shows a listing of the command file STATUS1:

Example 3-10. Command File for Status Information on X.25 Subdevices

```
== SCF command file displays status re: X25 subdevices,
== TCP/IP SUBNETs, and TCP/IP routes
allow errors
assume line $X250
status su (#tcpip1,#tcpip2,#tcpip3,#tcpip4,#tcpip5,#tcpip6)
assume process $ztc1
status
status subnet *
status route *
```

Configuration of an X.25 Connection for \NY

You must configure the NonStop TCP/IP subsystem for \NY to connect to a PDN X.25 network.

The set of six command files needed to configure \NY are given below. They are structured just like the command files for \LA.

- UP2 is the root level command file. It invokes the second level command files.
- SCFBASE2 starts SCP and MLMAN.
- X25SUBU2 sets the line attributes and adds SUs for line $X25A.
- TCPUP2 starts and configures NonStop TCP/IP and LISTNER processes. The only difference is the process names assigned ($ZTC2 and $LSN2 respectively).

  TCPUP2 invokes TCPSUBU2. TCPSUBU2 adds and starts the X.25 SUBNET and ROUTE objects to $ZTC2.
- STATUS2 displays status information about $X25A and $ZTC2.

Differences

The differences between this set of command files and the set for \LA are:

- In X25SUBU2:
  - The line name is $X25A.
  - The line attribute SRCADDR is 00000012300000.
- In TCPUP2:
  - The NonStop TCP/IP process name is $ZTC2.
  - The LISTNER process name is $LSN2.

  - The TELSERV process name is $ZTN2.
- In TCPSUBU2:
  - The SUBNET added specifies the device name $X25A and IP address 128.30.224.12
  - The LISTNER process name is $LSN2.
  - The ADDRMAP entry (#A2) has an IPADDRESS specification of 128.30.224.11 and an X121ADDR specification of 00000011300000.

UP2: The Main Command File

The command file UP2 acts as the root command file and invokes all the second-level subordinate command files.

Also included is a set of command files to take the subsystems down. Carefully examine each of these command files and verify the differences just listed. Example 3-11 shows a listing of the command file for UP2:

Example 3-11. Main Up File for X.25

```
== TACL command file to bring up TCP/IP over X25 for \NY.
== Set default volume and subvolume
   Volume $CB21.TCPIPX25
== Add and start PDN X25 subdevices
   scf/in X25SUBU2/
   delay 10 sec

== Start a TCP/IP Subsystem.
   command tcpup2

== Status everything
   scf/in status2/
```

X25SUBU2

The command file X25SUBU2 sets the line attributes and adds SUs for the line $X25A. Example 3-12 shows a listing of the command file for X25SUBU2:

Example 3-12. Command File for Adding X.25 Subdevices

```
== SCF command file to add and start X25 subdevices for TCPIP on
== \NY.
   allow all errors
   allow all warnings
== SWAN concentrator: Line $X25A available.
   assume line $x25A
   abort, sub all
   alter, netid x25xa,   srcaddr "00000012300000", Framemode &
   DTE, callingaddr on, SVCRANGE (1,128)
   add su #tcpip1,devtype (9,0),protocol ptp,recsize 1024
   add su #tcpip2,like #tcpip1
   add su #tcpip3,like #tcpip1
   add su #tcpip4,like #tcpip1
   add su #tcpip5,like #tcpip1
   add su #tcpip6,like #tcpip1
   start, sub all25A
```

TCPUP2

The command file TCPUP2 starts and configures NonStop TCP/IP and LISTNER processes ($ZTC2, $LSN2, and $ZTN2, respectively). Example 3-13 shows a listing of the command file for TCPUP2:

Example 3-13. Service Processes Up File for X.25

```
== TACL command file to bring up TCP/IP subsystem on \CB2
== Use HOSTS file for name resolution instead of DNS
        ADD DEFINE =TCPIP^HOST^FILE, FILE $CB21.TCPIPX25.MJHOSTS

== Initialize second TCP/IP process
        TCPIP /NAME $ZTC2, NOWAIT, PRI 180, CPU 0/

== Prior to starting LISTNER and TELSERV, use ADD DEFINE
statement
== to specify TCPIP process name
        ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC2

== Initialize TCPIP SUBNETs for X25
        SCF /IN TCPSUBU2/

== Initialize LISTNER process for FTPSERV, ECHOSERV, FINGSERV
        LISTNER /NAME $ZPORT, NOWAIT, PRI 160, CPU 1/&
                 PORTCONF

== Initialize TELSERV process
        PARAM TCPIP^PROCESS^NAME $ZTN2
        TELSERV/NAME $ZTN2,CPU 2,NOWAIT,PRI 170/-BACKUPCPU 3
```

TCPSUBU2

The command file TCPSUBU2 adds and starts the X.25 SUBNET objects and ROUTE objects to the NonStop TCP/IP process $ZTC2. The ADDRMAP specification maps the

IP address of \LA to its X121 source address, Example 3-14 shows a listing of the command file TCPSUBU2 (see Note on page 3-23 for information about obtaining the devicename in TCPSUBU2):

Example 3-14. Command File for Adding an X.25 Subdevice

```
== SCF command file to add a SUBNET to a TCP/IP process for an
== X25 line
   allow all errors
   assume process $ztc2
   alter subnet #loop0,ipaddress 127.1
   add subnet #pdn1,type x25,devicename $X25A, &
       ipaddress 128.30.224.12
   alter subnet #pdn1, subnetmask %HFFFFFF00
   add addrmap #a2, ipaddress 128.30.224.11,   &
      x121addr "00000011300000"
   start subnet *
   start route *
```

STATUS2

The command file STATUS2 displays status information about $X25A and the process $ZTC2. Example 3-15 shows a listing of the command file STATUS2:

Example 3-15. Command File for Status Information on X.25 Subdevices

```
== SCF command file displays status re: X25 subdevices,
== TCP/IP SUBNETs, and TCP/IP routes

   allow errors
   assume line $X25A
   status su (#tcpip1,#tcpip2,#tcpip3,#tcpip4,#tcpip5)

   assume process $ztc2
   status
   status subnet *
   status route *
```

Networking Services Provided in the Guardian Environment

Section 1, Configuration Quick Start and Example 3-1 through Example 3-16 provided configuration files for starting NonStop TCP/IP in the Guardian environment. This subsection focuses on the LISTNER process, the services that LISTNER provides, and the Domain Name Server (DNS). For information on the other protocols and services, see Appendix B, NonStop TCP/IP Processes and Protocols.

Configuring Subsystem Processes

This section describes the steps required to configure and start the various processes that operate the NonStop TCP/IP subsystem.

QIO

For NonStop systems, you do not need to configure or start the QIO subsystem; it is loaded and started during system cold load.

Configuring the NonStop TCP/IP Process

Once the NonStop system and the SLSA subsystem are running, start your NonStop TCP/IP processes.

To start a NonStop TCP/IP process, enter the following:

```
TCPIP /NAME process-name, &
       NOWAIT, &
       CPU cpu/backupcpu
```

Note. The TCP/IP process priority is set internally to 200 in this RVU.

SYS*nn* is the name of the system subvolume on which your NonStop TCP/IP software resides.

For *process-name*, enter a Guardian process name; the recommended form is $ZTC*x*, where *x* is one or more letters or numeric digits. The recommended name is $ZTC0; all NonStop applications look for this name.

Select the primary and backup *cpu*s from the Potential Access CPUs of the LIF. For information about why you must match the CPUs, see the Hint on page 1-7.

Finally, configure the NonStop TCP/IP SUBNETs and routes either through interactive commands issued through SCF or through programmatic commands issued by a management application.

A route is added automatically when you add a SUBNET, but you can add as many additional routes as your resources will allow. The route that is added automatically when you add the SUBNET has a name of the form #RT*. Additional routes are required when the remote network address portion of the IP address is not the same as that of the routes automatically added by the ADD SUBNET command. TCP/IP creates dynamic routes when it receives ICMP redirect packets. Dynamic routes have names of the form #DRT*. (When adding a route, note that the DSM object-naming conventions reserve #RT and #DRT; so select names that start with something else (such as #RO).) For a more detailed discussion of routing see *TCP/IP Illustrated* by W. Richard Stevens.

If you are using SCF, enter the sequence of commands shown in Example 3-16 for each subnet with which the NonStop TCP/IP process will communicate:

Example 3-16. Adding NonStop TCP/IP Subnets

```
3> SCF
-> ASSUME PROCESS process-name
-> ADD SUBNET subnet-name-1, &
->     TYPE ETHERNET, &
->     DEVICENAME lif-name-1, &
->     IPADDRESS ip-addr-1
-> ADD SUBNET subnet-name-2, &
->     TYPE ETHERNET, &
->     DEVICENAME lif-name-2,  &
->     IPADDRESS ip-addr-2
-> ADD SUBNET subnet-name-n, &
->     TYPE ETHERNET, &
->     DEVICENAME lif-name-n, &
->     IPADDRESS ip-addr-n
-> ALTER SUBNET #LOOP0, IPADDRESS 127.1
-> START SUBNET *
-> START ROUTE *
-> EXIT
```

process-name is the name you gave to the NonStop TCP/IP process when you started it through the RUN command.

subnet-name-n is the name to be assigned to the SUBNET.

lif-name-n is the logical interface (LIF) name used to communicate with the physical interface (PIF) on the ServerNet adapters connected to the system.

ip-addr-n is the Internet address of a host system.

The #LOOP0 SUBNET provides a loopback to the same host for conducting tests; by convention, its Internet address is always 127.1.

route-name-n is the name to be assigned to the route.

Additional routes are required when the remote network-address portion of the IP address is not the same as that of the routes automatically added by ADD SUBNET.

Starting the Domain Name Server (DNS)

Normally, you start the domain name server (NAMED) when you start NonStop TCP/IP. To start NAMED, enter the following TACL command:

```
TACL 3> RUN $SYSTEM.ZTCPIP.NAMED /NAME $ZNAME, PRI 140../  &
TACL 3> RUN $SYSTEM.ZTCPIP.NAMED /NAME $ZNAME, PRI 140../  &
TACL 3> [-p port-num] &
TACL 3> [[-b] boot-file-name]
```

NAMED listens to TCP and UDP port 53 for incoming requests. You can include the -p option in the RUN command to specify a different port number for use by NAMED. You must be certain that the port number is not the same as the number of any port specified for TELNET, FTP, or any other network program in use. If the port number is less than 1024, the domain name server can only be run by a user in the SUPER group.

The -b option specifies the boot file. You can omit the keyword -b and specify only the boot file name on the command line. The default boot file name is $SYSTEM.ZTCPIP.DNSBOOT.

Later versions of DNS, based on Berkeley Internet Name Domain (BIND) 9.*x,* are available on the NonStop server in the OSS environment. For information about these versions of DNS, see the *DNS Configuration and Management Manual*.

LISTNER Process

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

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

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

Note. If you do not specify a file name to be used in the PORTCONF file, the default assumed is $SYSTEM.ZTCPIP.PORTCONF.

Once started, LISTNER reads the SERVICES file to resolve the services configured in the PORTCONF file, and checks that the service name and corresponding port are valid. The SERVICES file contains information on the known services available on the Internet.

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

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

Each time a connection is made, connection-oriented services are provided by creating a new server process. The LISTNER process creates the server process to provide the requested service. The request may be received on a well-known TCP or UDP port. The server process then is passed an argument of the form sourceport.sourcehost, where sourceport is a decimal number and sourcehost is the source IP address in dotted decimal notation.

Data can pass between the target NonStop server and the remote client through the newly created socket until either the remote client or the server terminates the connection. For detailed information on configuring and starting the LISTNER process, refer to Section 1, Configuration Quick Start.

Starting the LISTNER Process

Once the NonStop TCP/IP process is started, you can start the LISTNER process. Only one LISTNER process can be run on each NonStop TCP/IP process.

To start the LISTNER process, enter a RUN or LISTNER TACL command as follows:

Example 1:

```
2> RUN $SYSTEM.SYSnn.LISTNER /NAME $ZPORT,in file,&
2> out file, term name, NOWAIT,CPU cpu, PRI 150/backupcu &
2> [portconf]
```

Example 2:

```
2> LISTNER /NAME $ZPORT, in file, out file, term name,&
2> CPU cpu, NOWAIT,PRI 150/backupcpu [portconf]
```

Example 1 shows the explicit RUN command. Example 2 shows the implicit RUN command.

SYS*nn* is the name of the system subvolume on which your LISTNER process resides.

cpu specifies the CPU numbers on which the LISTNER process and its backup should be run. You should run the LISTNER process on a different CPU than the one where the TCP/IP process runs.

Start the LISTNER process from a static or continuously available hometerm. If this condition is not possible, (as is the case when a dynamic terminal such as a Multilan dynamic window is used for startup operations), specify a local hometerm and IN and OUT files or specify $ZHOME (a terminal simulator) or use the virtual hometerm system, usually named $VHS.

You can enter the name of the PORTCONF file you have configured for the LISTNER process to use. The default PORTCONF file is $SYSTEM.ZTCPIP.PORTCONF. After modifying this file the LISTNER must be restarted. See PORTCONF File on page 3-40 for information on configuring the LISTNER process.

Note. The LISTNER process must be started by a user who has the user name SUPER.SUPER and the user ID 255,255 in order for a remote FINGER client user to access the information in the DOTPLAN and DOTPROJ files. Otherwise, the FINGER server cannot locate the user's default subvolume in which these files are located. For more information on the FINGER client and the DOTPLAN and DOTPROJ files, refer to the *TCP/IP Applications and Utilities User Guide*.

TELSERV Process

Once the NonStop TCP/IP process is started and the SUBNETs are configured, the next step is to start your TELSERV process. TELSERV provides the facilities to bring up terminal emulation (including TN6530) on the NonStop system. Refer to the *Telserv Guide* to start your TELSERV process. (Brief examples are included in the TCPIPUP files in Section 1, Configuration Quick Start. See also Appendix B, NonStop TCP/IP Processes and Protocols.)

File Transfer Protocol (FTP)

The File Transfer Protocol (FTP, RFC 959) facilitates the transfer of files between hosts. In the Guardian environment, the LISTNER process provides the FTP service.

Though a user can access FTP directly at a terminal, FTP is designed mainly for use by programs. The objectives of FTP, as stated in RFC 959, are:

1. Promote the sharing of files (computer programs and/or data)
2. Encourage the indirect or implicit (through programs) use of remote computers
3. Shield a user from variations in file-storage systems among hosts
4. Transfer data reliably and efficiently

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

An FTP session is initiated by entering the FTP run command on your NonStop system. FTP gives you a standard method for controlling data-transfer functions, connection establishment, connection management, and data-transmission modes.

FTP includes a standard set of commands and responses and, if applicable, reports corresponding state changes.

For a detailed description of the FTP server and client commands, refer to the *TCP/IP Applications and Utilities User Guide* For more detailed technical information on FTP, refer to RFC 959 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-739 through 2-808). Also, refer to the book *Internetworking with TCP/IP* (Douglas E. Comer, Prentice Hall, 1991).

FINGER Client/Server

The FINGER protocol provides a method for retrieving status information about one or all of the users on a particular system. This protocol is used over a TCP connection.

The process that handles incoming requests from FINGER clients is LISTNER. A line that has the proper information must be included in the PORTCONF configuration file for the LISTNER process, as described in Configuration Files in the Guardian Environment on page 3-34.

When the LISTNER process receives an incoming connection request for the FINGER program, it starts up the FINGER server program. The FINGER server is a TCP program that listens for connections on TCP port 79.

The FINGER server, once started, accepts the connection on a socket. It reads the socket to obtain a user name, then sends information about the user (or about all users, if no user name was specified).

For a detailed description of the operation of the FINGER server and client, refer to the *TCP/IP Applications and Utilities User Guide.* For more detailed technical information on FINGER, refer to RFC 742 (*DDN Protocol Handbook, Volume 2*, DDN Network Information Center, December, 1985, pp. 2-1017 through 2-1024). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

ECHO Client/Server

The ECHO protocol provides a debugging tool for determining access to network hosts. On the NonStop system, ECHO is used with TCP to allow an ECHO client to send data to the ECHO server. If access to the host on which the server is running is possible, the data will be returned to the requesting client. If the data is not returned to the client, the named host is inaccessible or is not running the specified protocol.

The ECHO client can be run from any terminal connected to the NonStop server, including a workstation emulating a 6530 or network virtual terminal. NonStop TCP/IP ECHO does not service UDP ports.

For a detailed description of the ECHO server and client, refer to the *TCP/IP Applications and Utilities User Guide*. For more detailed technical information on ECHO, refer to RFC 862 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-1001 through 2-1002). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

Configuration Files in the Guardian Environment

This section describes the files you can customize to configure your Internet environment.

Configuration Files for the Internet Environment

There are several files that come on your system update tape (SUT) that are used for configuring your internet environment. You need to customize some of these files to match your own NonStop TCP/IP environment.

- Customize the first three files here for your internet:
    - HOSTS file
    - RESCONF file
    - NETWORKS file
    - PROTOCOL file
    - SERVICES file
    - PORTCONF file
- The Domain Name Server data files:

Note. Later versions of DNS, based on Berkeley Internet Name Domain (BIND) 9.x, are available on the NonStop server in the OSS environment. For information about these versions of DNS, see the *DNS Configuration and Management Manual*.

-
    - DNSBOOT
    - DNSCACHE
    - DNSLOCAL
    - DNSHOSTS
    - DNSREV
- The SMTP configuration file: SMTPCONF

The names of all these files begin with $SYSTEM.ZTCPIP, unless they are installed on some other subvolume.

These files are standard EDIT type files (file code 101); you can use a text editor such as EDIT or PS Text Edit (TEDIT) to modify them. Each of these files consists of a sequence of one-line entries. Lines starting with a pound sign (#) in the Internet environment and SMTP configuration files listed are comment lines. Comment lines start with a semicolon (;) in the Domain Name Server (DNS) data files.

HOSTS file

NonStop TCP/IP hosts communicate through their IP addresses. However, IP addresses are not easy to remember. As a result, it is common practice to assign host names to IP addresses. Hosts then can be referred to by name. To provide for the translation, a Domain Name Resolver (DNR) is provided. A DNR, in turn, uses either a Domain Name Server (DNS) or a HOSTS file; configure the DNR to use one or the other.

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_address 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.

Consider the example that follows:

```
# HOSTS file
127.0.0.1    me loop geoff mark cyclone
# \CB2 is the gateway between subnets for \ENC1 and \CB1
128.1.1.1    CB21  cb21  # on subnet 1.0, lan01 08008E0002A6
128.1.2.1    CB22  cb22  # on subnet 2.0, lan02 08008E000B2D
#
```

Notice that the first entry beginning with 127.0.0.1 has several aliases. This means that you can use any of these aliases to communicate with the destination host that has the IP address 127.0.0.1.

Note. The IP address 127.0.0.1 is a TCP/IP convention that refers to “this” host or loopback.

The alias for a host is optional, and a host can have more than one alias; however, the aliases must be separated by spaces and be on the same line. The other entries (beginning with 128.1.1.1) each have two host names: one in uppercase and one in lowercase. Neither the host name nor the alias is case sensitive.

You can add comments to the HOSTS file by preceding the comment with a pound sign (#). You can add comments as separate lines of the file or after the IP address and host entry. Include comments like the line name or hardware address that can be used for reference.

As indicated earlier, you must configure the DNR to use a HOSTS file; otherwise a 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.tcpipSF.hosts
```

Remember, 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 a DNS and consults a RESCONF file.

Also, note that you 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 is executed automatically every time you log on to the NonStop system.

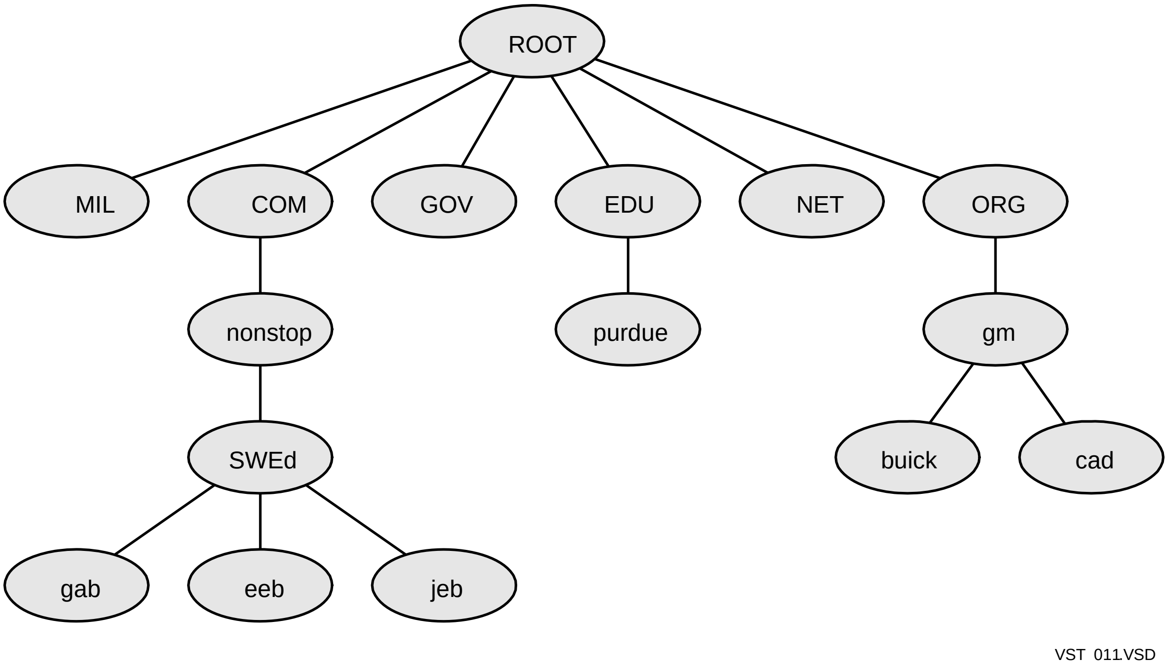
RESCONF File

The Domain Name Resolver (DNR) resolves domain names to IP addresses using either a HOSTS file or a DNS to provide the translation. If you did not set the TCP^HOST^FILE parameter, the DNR assumes it must use a DNS. To determine which DNS to use, the DNR interrogates the RESCONF file. Thus, you must configure the RESCONF file when using a DNS. Also, note that the NonStop TCP/IP sockets library provides a procedure call to the DNR.

Domain Name System

The Domain Name System is a hierarchical naming system consisting of domains and subdomains. The domain immediately above the current domain is called the parent. The TCP/IP Domain Name System structure is illustrated in Figure 3-5.

Figure 3-5. Domain Name System Hierarchy



It is important to remember that the Domain Name System is based on administrative boundaries and not the topology of the network. The Internet Network Information

Center (NIC) has authority over the root and the first level domains (for example, MIL, COM, GOV, EDU, NET, ORG). NonStop received permission from the NIC to add the subdomain NonStop. The network administrator at HP, however, now takes responsibility for all the subdomains that are created at NonStop.

Domain names are written with the most specific name first and the most general last, for example:

```
eeb.SWEd.nonstop.com
```

In this example, com is the highest level domain while eeb is the lowest level domain. Consequently, it is impossible to use the domain name syntax to distinguish host names from domain names.

By analogy, in the telephone network, each telephone number is divided into an area code, a prefix, and the telephone number. Because most telephone calls are local, the calls are not routed outside the local area unless they are destined for another area code. This means that the switches for routing calls outside the local area are not burdened with the traffic of local calls.

In the same way, most Internet traffic is local and the name resolution can be handled by the local Domain Name Server. Only when there are non-local names that require resolution are non-local servers accessed. A by-product of this condition is that instead of addressing a local destination as

```
eeb.SWEd.nonstop.com
```

you can address it as

```
eeb
```

Domain Name Servers also are required to know the address of the parent server, that is, the address of the server in the parent domain. Also, all the information of one server is to be replicated in another server on the same level with no single point of failure. This arrangement provides a backup in the event that one server fails.

RESCONF Details

The Domain Name Resolver (DNR) uses the information in the RESCONF file to access the Domain Name Server (DNS). The RESCONF file contains the name of the domain in which the current host is running (domain NonStop.COM). The following example lists three DNS IP addresses (nameserver 127.1, nameserver 50.0.0.23, and nameserver 50.0.0.36):

```
domain NonStop.COM
nameserver 127.1
nameserver 50.0.0.23
nameserver 50.0.0.36
```

The first server address is the primary server. If that server is unavailable, the resolver contacts the second server. If the second is unavailable, the resolver tries to contact the third server. Notice that the first server address is 127.1; this is the address you would use if the current NonStop host had a server available. The RESCONF file that comes with your NonStop TCP/IP software is a prototype; you should customize this file.

To override the default RESCONF file, use the environment variable TCPIP^RESOLVER^NAME. This variable provides flexibility in selecting the RESCONF file accessed for name resolution. The default RESCONF file is located on $SYSTEM.ZTCPIP. The following example contains a TACL ADD DEFINE command you can use to select a different RESCONF file:

```
2> ADD DEFINE =tcpip^resolver^name, FILE $data.user.resconf
```

Using the Domain Name resolver is the preferred way of resolving names on the network. If a name server is not available on the network, use a HOSTS file, as described earlier.

NETWORKS File

The NETWORKS file lists the names, numbers, and aliases of networks known to the current host. This file converts an Internet network address to a symbolic name. Applications use this file when they call a getnetbyaddr or a getnetbyname function. The NETWORKS file included with the NonStop TCP/IP software is a prototype; you should customize this file.

Each entry of the NETWORKS file has the following format:

```
network_name  network_number [alias...]
```

where network_name is an alphanumeric name, network_number is the assigned network number, and alias is an alphanumeric name.

The alias is optional for each network, and each network may have more than one alias. The aliases for each network must be on the same entry line and separated by spaces.

Example 3-17 shows the NETWORKS file. The pound sign (#) indicates a comment line.

Example 3-17. NETWORKS File

```
# customer networks
#
loopback        127        testnet
sun-ether       192.9.200  sunether ethernet
sun-oldether    125        sunoldether
#
# Internet networks
#
arpanet         10              arpa
ucb-ether       46              ucbether
```

PROTOCOL File

The PROTOCOL file contains the names of the protocols currently supported by the NonStop TCP/IP software as well as some not currently supported. Applications use the PROTOCOL file to get protocol names and Internet protocol numbers. When an application calls the functions getprotobyname or getprotobynumber, the PROTOCOL file provides this information. You do not need to alter this file.

Each entry has the following format:

```
protocol_name  protocol_number  PROTOCOL_NAME
```

where protocol_name is the protocol name in lowercase, protocol_number is the well-known Internet protocol number and PROTOCOL_NAME is the protocol name in uppercase. (Because the functions getprotobyname and getprotobynumber are case sensitive, both uppercase and lowercase representations of the protocol names are included in the PROTOCOL file.)

Example 3-18 shows the PROTOCOL file (precede comment lines with a pound sign (#)):

Example 3-18. PROTOCOL File

```
# @(#)protocols 1.1 NonStop
#
# Internet (IP) protocols
#
ip      0     IP      # internet protocol, pseudo protocol
number
icmp    1     ICMP    # internet control message protocol
ggp     3     GGP     # gateway-gateway protocol
tcp     6     TCP     # transmission control protocol
pup     12    PUP     # PARC universal packet protocol
udp     17    UDP     # user datagram protocol
```

SERVICES File

The SERVICES file contains the Internet port level services that are available with the NonStop TCP/IP software. Applications use the SERVICES file to get the service port numbers and service names. When the application calls the getservbyname or getservbynumber function, TCP/IP uses the SERVICES file to provide that information. You do not need to alter this file.

Example 3-19 shows the SERVICES file:

Example 3-19. SERVICES File

```
# Network services, Internet style
echo            7/udp
discard         9/udp           sink null
systat          11/tcp
daytime         13/tcp
netstat         15/tcp
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
whois           43/tcp          nicname
# usually to sri-nic
domain          53/udp
domain          53/tcp
hostnames       101/tcp         hostname
# usually to sri-nic
sunrpc          111/udp
sunrpc          111/tcp
# Host specific functions
tftp            69/udp
rje             77/tcp
finger          79/tcp
```

Each entry specifies a service name, the port number through which that service is accessed, and the corresponding protocol which supports that service. You can use an alias to identify the service.

PORTCONF File

The PORTCONF file specifies the ports that the LISTNER process listens to and the corresponding server program it invokes when the request comes in. You do not need to alter this file.

Example 3-20 shows the PORTCONF file.

Example 3-20. PORTCONF File

```
#
ftp     $system.ztcpip.ftpserv
finger  $system.ztcpip.fingserv
7       $system.ztcpip.echoserv
```

The FTP, ECHO, FINGER and SMTP servers are not running all the time. The LISTNER process starts these servers when a request comes in from a client process. This LISTNER process needs to know which ports to listen to; to do so, the LISTNER process cross-references the information in the PORTCONF file and the SERVICES file.

Simple Mail Transfer Protocol (SMTP)

The Simple Mail Transfer Protocol (SMTP, RFC 821 and RFC 1495) gateway implements the SMTP on NonStop systems running Guardian. The SMTP gateway uses the sockets interface to establish connection with Internet hosts. It provides an interface between TRANSFER mail systems and the TCPIP process.

The SMTP processes (SMTPRCV and SMTPSND), first open the TISERV (Transfer Process) (specify up to five TISERV process names). The SMTP processes starts a session with TRANSFER by logging in as the SMTP MAIL gateway correspondent (SMTPGATE). Once the processes log into TRANSFER, they can make requests to TRANSFER to store, retrieve, or delete messages. Specify the mail correspondent name and password in the SMTPCONF file, as described Option processing—O on page 3-51.

The default location for the SMTPCONF file is $SYSTEM.ZTCPIP.SMTPCONF; however, the SMTPSND and SMTPRCV programs can be given the -C option to use a different location for SMTPCONF.

The SMTPRCV process, which handles mail arriving from the network, is started by the LISTNER process. The LISTNER process listens for incoming service requests on well-known ports. The well-known port number for SMTP services is TCP port 25. This is set up in the PORTCONF configuration file for the LISTNER process by adding an entry for SMTP.

The SMTPSND process is started by the system administrator as part of the NonStop TCP/IP startup mechanism. This process can be run from TACL.

Setting Up SMTP

To set up the SMTP gateway (SMTPGATE), do the following:

1. Edit a copy of the supplied SMTPCONF file to make the required changes. Refer to Setting Up the Configuration File, later in this section, for detailed instructions on making these changes.
2. Set up an entry for the SMTPRCV in the file $SYSTEM.ZTCPIP.PORTCONF for the LISTNER process, as follows:

   ```
   SMTP     $SYSTEM.ZTCPIP.SMTPRCV
   ```

   If you are using a different location for SMTPCONF (other than $SYSTEM.ZTCPIP.SMTPCONF), then the entry in the PORTCONF file needs to be as follows:

   ```
   SMTP $SYSTEM.SYSTEM.SMTPRCV -Cconfiguration-file
   ```

   Note. The -C syntax for SMTPRCV differs from the -C syntax for SMTPSND; do not enter a space between -C and *configuration-file* for SMTPRCV.

3. Run the file $SYSTEM.ZTCPIP.SMTPRCV including the runtime option -bt, and specify the configuration file you created in step 1 using the run-time option -C *configuration-file*.

   ```
   RUN $SYSTEM.ZTCPIP.SMTPRCV -bt -Cconfiguration-file
   ```

   Note. The -C syntax for SMTPRCV differs from the -C syntax for SMTPSND; do not enter a space between -C and *configuration-file* for SMTPRCV.

   This command prints any errors that are present in the configuration file you created. Fix these errors and repeat this step until no more errors are present.

4. Create the SMTP gateway correspondent that you specified in the configuration file.

5. Run the SMTPSND program. TMF, TRANSFER, PSMAIL, PATHWAY, and SMTPCONF should be set-up before invoking SMTPSND. The SMTPSND process can be run from TACL as shown:

   ```
   RUN $SYSTEM.ZTCPIP.SMTPSND /CPU xx, PRI xx/  &
            [-time timevalue] [-C configuration-file]
   ```

   Note. The -C syntax for SMTPSND differs from the -C syntax for SMTPRCV; you must include a space between -C and *configuration-file* for SMTPSND.

   The -time parameter specifies the time (in minutes) that the SMTPSND sleeps between scanning its INBOX and scanning its RETRYFOLDER. The maximum value is 32K -1, and the minimum value is 0. If an error occurs in specifying this parameter, the default value (5 minutes) is used. The priority of the SMTPSND process may be low; the recommended range is 140 to 150.

6. Restart the LISTNER program if NonStop TCP/IP is already running.

Note. Only the statements listed in item 1 should be changed or altered by the administrator. Altering any other lines requires thorough understanding of the effect of the changes, many of which are beyond the scope of this manual.

Setting Up the Configuration File

The basic intelligence in the SMTPSND and SMTPRCV is the address parsing mechanism. This mechanism is driven from the configuration file, $SYSTEM.ZTCPIP.SMTPCONF. This file is set up so that it can parse all legal RFC 822 addresses.

The SMTP configuration file (SMTPCONF) is an EDIT file that you can modify. The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines. Blank lines and lines beginning with a pound sign (#) are comments.

Required Edits

To set up SMTP for your organization, you only need to make a few changes to the SMTPCONF file. The editing required includes the following:

- Change the Class W (Cw) specification (Internet host-name) to your host-name.
- Change (DN) to the hostname that you want other hosts to know you as.
- Alter the macro A (DA) to add in the ARPANET gateway, if you have one.
- Alter the Class A (CA) to your ARPANET gateway, if you have one.
- Define your local domain class (CD line).
- Define your major UUCP relay host class R (CR line).
- Alter the macro R (DR) to the major UUCP relay host name.
- Alter the option OC to your SMTP gateway correspondent name.
- Alter the option OP to the password for your SMTP gateway correspondent. Remember that the password is case-sensitive.
- Alter the option OA to the system administrator who should receive mail that cannot be returned or delivered.
- Alter the option OT to specify the names of TISERV processes if ones other than the default $ZCC0 are to be used.

Optionally, you can alter the option OE if a message expiry (expiration) time other than 96 hours is required.

Note. If you make changes to the configuration file after the SMTP server is started, you must restart it before the changes to take effect. Restarting the SMTP server or client does not cause any loss of messages.

Example of SMTPCONF File Lines

The following text is an example of the SMTP configuration file supplied with the NonStop TCP/IP software. These lines indicate the changes you must make to the file.

```
#############################################################
#####                                                    ####
#####           TRANSFER-SMTP CONFIGURATION FILE         ####
#####                                                    ####
#############################################################

##################
#   local info   #
##################

#
# Enter the internet host-name for your machine here.  In
# this example, the host-name is “Expand”. Replace “Expand”
# with your machine name.
```

```
#
Cwexpand localhost me

#
# This host-name is specified in the format by which you
# want the outside world to know your host.  Sometimes
# there is confusion in the way the $w.$D macro is
# represented, If your host-name already contains your
# domain name.  Host-names are obtained from the $ZTC0
# process.  Check to see if the domain name is a
# part of it.  In any case, define the host-name here
# in full format.  For example, HOST.DOMAIN
#
DNexpand.NonStop.COM
#
#
# Enter the name of the ARPANET gateway here.  The gateway
# shown below is gate.NonStop.COM.  Replace it with yours.
#
DAgate.NonStop.COM
CAgate.NonStop.COM

#
# The Local-Domains go here.  These are usually
# Company-Name.COM or University.EDU.  Check with your
# network administrator if you do not know your domain
# name.  In this example, the domain name is NonStop.COM.
# Change it to your domain name.  For example, UCLA.EDU,
# NASA.GOV, or ATT.COM.
#
DDNonStop.COM
CDNonStop
#
# Main top level domain.  In this example, COM is the TOP
# level domain.  LOCAL is also another domain.  This allows
# for receiving mail addressed to user@host.LOCAL or
# user@host.LOCAL.COM.  Change COM to the name of your top
# level domain.
#
CTCOM LOCAL
#
# The following is the major UUCP relay.  This is required
# to convert addresses such as uucphost!user to the form
# @relay:user@uucphost.UUCP.  It also converts names of
# the form user@uucphost.UUCP to @relay:user@uucphost.UUCP.
# In this example, the relay machine is gate.nonstop.COM.
#
DRgate.nonstop.COM
CRgate.nonstop.COM
#
# Enter the Transfer correspondent name here.  In this
# example, the correspondent name is SMTP.
#
OCSMTP
#
```

```
# Enter the Transfer correspondent password here.  Replace
# <password> with your password.  Note that the password is
# case sensitive, so be careful about mixing uppercase and
# lowercase in the password.
#
OP<password>
#
# Replace <sys-admin> with the name of the System
# Administrator to whom undeliverable mail is forwarded.
# The default name is SYS-ADMIN.
#
OA<sys-admin>
#
# Define the names of TISERV processes if you use ones
# other than $ZCC0.  Remember that $ has to be escaped by
# another $ to prevent expansion.  More than one (up to 5)
# of these lines may be specified.
#
OT$$ZCC0
OT$$ZCC1
OT$$ZCC2
OT$$ZCC3
OT$$ZCC4
#
# To enable distribution lists, remove the comment from
# the next option line (ODY). A comment symbol disables an
# option line.
#ODY
```

You should not make other changes to the configuration file, unless you have a complete understanding of how the rules are applied to parse addresses. If you need to make such changes, read the following rule line syntax carefully.

Rule Line Syntax

The first character on a line specifies the purpose of the line:

R Rule for rewriting

S Rule set number

D Define macro

C Item class

O Option processing

H Header fields

M Define mailer

# Comment line

The following paragraphs define each semantic identifier.

Rule for rewriting—R

The core of address parsing is the rewriting rules, which you specify in lines starting with R. These rules are ordered production systems. Each rule has a left-hand side and a right-hand side. When an address is to be rewritten, the address parser scans through the rule sets comparing the left-hand side against the specified address. When a rule matches, the address is replaced by the right-hand side of the rule. The syntax of an R line is as follows:

`RLHS` *`space`* `RHS` *`space`* `COMMENTS`

In the syntax, *`space`* denotes one or more spaces that separate the left-hand side from the right-hand side and the comments. If a space has to be embedded in the rules, precede it with a backslash (\) character. Any backslash character found in the input stream is ignored and the next character is taken literally (as part of the current token being scanned).

SMTP includes several sets of rewriting rules. Some are used internally and must have specific semantics. Others do not have specific semantics and can be referred to by the mailer definitions or by other rewriting sets.

The left-hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced by using a dollar sign ($).

The metasymbols are:

`$*`	Match zero or more tokens
`$+`	Match one or more tokens
`$-`	Match exactly one token
`$=x`	Match any token in class x
`$~x`	Match any token not in class x

If any of these metasymbols match, they are assigned to the symbol $n for replacement on the right-hand side, where n is the index in the left-hand side. The first piece is numbered 1. For example, if the following left-hand side:

```
$+:$+
```

is applied to the input:

```
UCBARPA:eric
```

the rule will match, and the values passed to the right-hand side are:

```
$1  UCBARPA
$2  eric
```

When the left-hand side of a rewriting rule matches, the input is deleted and replaced by the right-hand side. Tokens are copied for rewriting directly from the right-hand side unless they begin with a dollar sign.

Tokens beginning with a dollar sign are expanded according to the following metasymbols:

`$n`	Substitute indefinite token n from the left-hand side
`$[name$]`	Canonicalize name
`$>n`	Call rule set n
`$@host`	Specify host
`$:user`	Specify user
`$#mailer`	Specify mailer

The $n syntax substitutes the corresponding value from a $+, $*, $=, or $~ match on the left-hand side. You may use $n syntax anywhere.

A host name enclosed between $[and $] is looked up using the sockets library routines and replaced by the canonical name. For example, $[csam$] would become lbl-csam.arpa and $[[128.32.130.2]$] would become vangogh.BigCityU.edu.

The $>n syntax causes the remainder of the line to be substituted as usual, then passed as the argument to rule set n. The final value of rule set n then becomes the substitution for this rule.

For example:

```
R@$+,$+:$+      @$1:$2:$3       # change all "," to ":"
R@$+:$+         $@$>6<@$1>:$2   # src route is canonical
```

The second line indicates that rule set 6 must be used next.

Use the $# syntax only in rule set zero. $# syntax causes rule-set evaluation to terminate immediately and signals to the mailer that the address has been completely resolved. The complete syntax is as follows:

```
$#mailer$@host$:user
```

This syntax specifies the {mailer, host, user} information necessary to direct the mailer. If the mailer is local, you may omit the host part. The mailer and host must be a single word but the user may consist of multiple parts.

A right-hand side may be preceded by a $@ or a `$:` to control evaluation. A $@ prefix causes the rule set to return with the remainder of the right-hand side as the value. A $: prefix terminates the rule immediately but sets the rule set to continue. Use this prefix to avoid the continued application of a rule. The prefix is stripped before continuing.

The $@ and $: prefixes may precede a $> specification. For example:

```
R$+       $:$>7$1
```

matches anything, passes what it matches to rule set seven, and continues. The $: is necessary to avoid an infinite loop.

Substitution occurs in the order described; parameters from the left-hand side are substituted, hostnames are canonicalized, subroutines are called, and, finally, $#, $@, and $: are processed.

Rule set number—S

The rules (R) are combined into rule sets, as specified by the S command. The syntax of this command is as follows:

S*n*

The S command sets the number of the current rule set being collected to *n*. If you begin a rule set more than once, the old definition is deleted. The maximum number of rule sets that can be defined is 30.

The above syntax defines a rule for the rule set *n*, as defined above. The rule set can consist of server rules which are applied in sequence until a match is found.

The fields must be separated by at least one space character. Embedded spaces may appear in the fields only if they are preceded by the backslash character (\). The left-hand side is a pattern that is compared to the input. If the pattern matches the input, the input is rewritten to the right-hand side. The comments are ignored.

The following example defines rule set 6 (S6):

```
S6
R$-%$-.TRANSFER        $1#$2         user%expand.Transfer
R$-%$-.TRANSFER@$w     $@$1#$2       user%expand.Transfer
```

An address of the following form is matched by the second rule in the rule set:

```
kent_joe%ab5.TRANSFER@pubs.kentcomm.com
```

where:

```
$-%$- matches kent_joe%ab5
```

and

```
$w matches pubs.kentcomm.com.
```

The address is rewritten as:

```
kent_joe#ab5
```

Define macro—D

Macros are named with a single character. You can use the entire ASCII set; however, select user-defined macros from the set of upper-case letters only. Lower-case letters and special symbols are used internally.

The syntax for macro definitions is as follows:

D*xval*

where *x* is the name of the macro and *val* is the value of the macro.

Macros are interpolated using the construct $*x* where *x* is the name of the macro to be interpolated. In particular, lower-case letters are reserved for specifying special semantics which are passed information in or out of smtpgate. Some special characters are reserved for specifying conditionals.

Specify conditionals by using the following syntax:

```
$?x text1 $| text2 $.
```

This syntax interpolates `text1` if the macro $*x* is set, and `text2` if it is not; you can omit the else clause $|.

You must define the following macros:

e The SMTP entry message. This message is generated in the 220 greeting line, first displayed when a connection is made to the SMTPRCV.

j The official domain name for this site, which is the string derived from the gethostname call (also the name of the host that is set with the SCF command ALTER, HOSTNAME).

The name of the mailer daemon (for error messages). This is the name used as the sender's name when error messages are sent out by the SMTPS server. This name should generally be POSTMASTER.

o The set of operators in addresses. This macro is used in address parsing to separate tokens in the rule sets. It generally contains the following set of characters:

```
<>[].@%!^=
```

q The default format of the sender address.This macro specifies how the From: line looks in the sender's address.

In addition, the following macros are internally defined; do not redefine them:

a The origination date in ARPANET format

b The current date in ARPANET format

c The hop count

d The date in Guardian format

f The sender (from) address

g The sender address relative to the recipient

h The recipient host

r The protocol used

s The sender's host name

t A numeric representation of the current time

u The recipient user

`v` The version number of SMTPSND

`w` The hostname of this site

`x` The full name of the sender

The $e macro is displayed when SMTP starts up, that is, when connection is first established with the mail RECEIVER. The first word must be the $j macro. The $j macro should be in RFC 821 format. The $n macro can be considered constant. The $o macro consists of a list of characters that are considered tokens and that separate tokens when SMTP parses addresses. For example, if @ were in the $o macro, then the input a@b would be scanned as three tokens: a, @, and b. Finally, the $q macro specifies how an address should appear in a message when it is defaulted.

For example:

```
# my name
DnSMTP-MAILER
#
# Delimiters in an address (operators)
#
Do.:%@!^=/[]
# format of a total name, eg:  john@host (John Smith)
Dq$g$?x ($x)$.
# SMTP login message
De$j SMTPGateway $v/$V ready at $b
```

A null string is used for both macros and classes, if they are not defined. For example, the following string evaluates to .COM if $F is not defined:

```
$F.COM
```

The use of $F does not cause an error.

Item Class—C

You can define classes of words to match on the left-hand side of rewriting rules. For example, you could create a class of all local names for this site so that attempts to send to oneself can be eliminated. Use upper-case letters in class names; lower-case letters and special characters are reserved for internal mailer use.

The syntax is:

```
Cxword1 [ word2 [ word3 ... ]]
```

This syntax defines the class x to match any of the named words. You may split the words among multiple lines. For example:

```
CHvlx commvlx
```

and

```
CHvlx
CHcommvlx
```

are equivalent.

Option processing—O

You can set many options from a configuration file. Options are represented by single characters. The syntax of this line is as follows:

```
Oxstring
```

This syntax sets option x to the value specified in string. The options are:

- C**—**Mail Correspondent Name. The syntax of this line is as follows:

  ```
  OCSMTP_MAILER
  ```

  This option sets the mail correspondent name to SMTP_MAILER. If you do not specify the correspondent name, the default is SMTPGATE.

- P**—**Mail Correspondent Password. The syntax of this line is as follows:

  ```
  OPSMTP_PASS
  ```

  This option sets the password for the SMTPGATE to SMTP_PASS. This option has no default.

- A**—**Mail Administrator. The syntax of this line is as follows:

  ```
  OAKENT_JOE
  ```

  This option sets the System Administrator to KENT_JOE. This is the person who gets the mail messages that cannot be delivered and cannot be returned. If a name is not specified, the default is SYS-ADMIN.

- T**—**Set TISERV name. The syntax of this line is as follows:

  OT$ZCC*x*

  This option line sets the name of a TISERV process to $ZCC*x*, where *x* can be a digit from 0 to 9. You can specify up to five TISERV process names. If you do not specify T1SERV, the default process name is $ZCC0.

- E - Set message expiry time. The syntax of this line is as follows:

  ```
  OE72
  ```

  This option sets the period (in hours) that the SMTP gateway attempts to transmit a message to a destination host. The default value is 96 hours (4 days).

  The SMTP gateway keeps trying to reach a host until that time expires. If the host is unreachable after that period, the message is returned to the sender with a notification that the original mail message to a specific user has expired. If a negative value or a value larger than 32K -1 is specified, the default value of 96 hours is used.

Header fields—H

This line is used to define the header of messages. The syntax is as follows:

H[?*mflags*?] *header-name*:*header-template*

Continuation lines are reflected directly into the outgoing message.

The *header-name* specifies the name of the header. You can use (but are not limited to) the following header names:

From

Subject

Resent-From

Return-Path

Date

Received

Resent-Date

Message-Id

The *header-template* is macro-expanded (processed by macros) before insertion into the message.

If you specify the *mflags* (surrounded by question marks), you also must state at least one of the specified flags in the mailer definition for this header to be automatically output. If one of these headers is in the input, it is reflected to the output regardless of these flags.

For example, for the following line to be output in the header, the mailer defined by the mailer definition (see Define Mailer—M below) should contain the flag P:

```
H?P?Return-Path: $g
```

This header definition typically returns a line such as:

```
Return-Path: KENT_JOE@pubs.kentcomm.com
```

Sample header lines are:

```
H?P?Return-Path: $g
HReceived: $?sfrom $s $.by $j ($v/$Z)
        id $i; $b
H?D?Date: $a
H?F?From: $q
H?x?Full-Name: $x
HSubject:
H?M?Message-Id: <$t.$i@$j>
```

The header of a message processed by such a header line could be:

```
Return-Path: KENT_JOE@pubs.kentcomm.com
Received: by pubs.kentcomm.com (5.52/1.14.1.1)
        id AA16145; 2 Nov 89 14:24:13 -0800
Date: Wed, 1 Nov 89 20:54:31 PST
From: SMTPGATE(KENT_JOE)
Subject: Test of headers in message
Message-Id: <8911020454.AA00352@pubs.kentcomm.com>
```

Notice that the line containing the Full-Name did not appear here, because the mailer line did not contain the x flag in the flags specified.

Define Mailer—M

Each mailer must have an internal name. This name can be arbitrary, except that the names local must be defined.

The format of a define mailer line is:

```
M mailer-name field=value [,field=value ...]
```

where *mailer-name* is the name of the mailer (used internally only) and the *field=value* pairs define attributes of the mailer. Fields are:

P - (Path)	The mailer path.
F - (Flags)	Special flags for this mailer.
S - (Sender)	A rewriting set for sender addresses.
R - (Recipient)	A rewriting set for recipient addresses.
E - (End-of-Line)	The end-of-line string for this mailer.
M - (Maxsize)	The maximum message length to this mailer.
A - (Arguments)	Arguments to be passed to this mailer.

You must give the actual name of the mailer in the P field; however, this mailer is accessed through an IPC connection, use the string [IPC] instead. The flags field influences the header field: the headers are expanded according to the flags specified here. If you use the ?x? format in the header field, that header line is included if you defined the flag x in this flag string.

For example, the following line indicates that a TCP connection needs to be established to the host specified in $h:

```
Mtcp,   P=[IPC], F=mPSFMueXLC, S=14, R=24, A=IPC $h, E=\r\n
```

The S and R fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses, respectively. These sets are applied after the sending domain is appended and the general rewriting sets (numbers one and two) are applied, but before the output rewrite (rule set four) is applied.

For example:

```
Mlocal, P=TRANSFER
```

indicates that local mail has to be delivered to TRANSFER.

Issues

RFC 1495 describes the format of the mail message itself. The parser in the address parser follows this RFC closely, to the extent that many standards described in this document cannot be changed without changing the code. In particular, the characters < > ( ) “ \ all have special interpretations. Any attempt to use these characters for something other than their RFC 1495 purpose in addresses can produce undefined results.

RFC 819 describes the specifics of domain-based addressing. These specifics are briefly described in RFC 822 as well. Essentially, each host is given a name that is a right-to-left, dot-qualified pseudo-path from a distinguished root. The elements of the path need not be physical hosts; the domain is logical. For example, at BigCityU one legal host may be a.CC.BigCityU.EDU; reading from right to left, EDU is a top-level domain comprising educational institutions, BigCityU is a logical domain name, CC represents the Computer Center (in this case a strictly logical entity), and a is a specific host in the Computer Center.

Errors in Configuration Files

In normal operation of the SMTPSND and SMTPRCV processes, errors found in the configuration file are silently ignored. However, you can verify the address rewriting mechanism of the configuration file by using the SMTPRCV for testing, as follows:

```
RUN $SYSTEM.ZTCPIP.SMTPRCV -bt -Cconfiguration-file
```

The -bt flag tells the SMTPRCV that being used to test the configuration file specified by -C configuration-file. The default SMTP configuration file $SYSTEM.ZTCPIP.SMTPCONF is tested if a different file is not specified. This command prints any errors that are present in the configuration file you specify.

If the configuration file contains no errors, you are prompted for the rule set to use and an input address to be parsed.

```
ADDRESS TEST MODE
Enter <ruleset> <address>
>
```

Enter a line similar to the following:

```
> 0 user@sun.kci
```

This entry specifies that rule set 0 is to be applied to the address `user@sun.kci` after it is parsed.

Examples of Address Parsing

Rule set three is always applied first. Rule set three should turn the address into canonical form which have the following basic syntax:

```
local-part@host-domain-spec
```

Rule set zero is applied after rule set three to addresses that actually specify recipients. The address must resolve to {mailer, host, user}. You must define the mailer in the mailer definitions from the configuration file.

The parsing rules are applied as follows:

```
rule set  3   input: "user" "@" "sun" "." "kci"
rule set  8   input: "user" "@" "sun" "." "kci"
rule set  8 returns: "user" "@" "sun" "." "kci"
rule set  6   input: "user" "<" "@" "sun" "." "kci" ">"
rule set  6 returns: "user" "<" "@" "sun" "." "kci" ">"
rule set  3 returns: "user" "<" "@" "sun" "." "kci" ">"
rule set  0   input: "user" "<" "@" "sun" "." "kci" ">"
rule set  3   input: "user" "@" "sun" "." "kci"
rule set  8   input: "user" "@" "sun" "." "kci"
rule set  8 returns: "user" "@" "sun" "." "kci"
rule set  6   input: "user" "<" "@" "sun" "." "kci" ">"
rule set  6 returns: "user" "<" "@" "sun" "." "kci" ">"
rule set  3 returns: "user" "<" "@" "sun" "." "kci" ">"
rule set  6   input: "user" "<" "@" "sun" "." "kci" ">"
rule set  6 returns: "user" "<" "@" "sun" "." "kci" ">"
rule set  0 returns:
    "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com" "^X" "user"
                            "<" "@" "sun" "." "KCI" ">"
```

Rule set 0 has reduced the address to the canonical form {mailer, host, recipient}, where:

```
mailer      = tcp
host        = kciwrl.kci.com
recipient   = user<@sun.KCI>
```

```
Mtcp,   P=[IPC] S=14, R=24, A=IPC $h
```

rule set 24 input:

```
"^V" "tcp" "^W" "kciwrl" "." "kci" "." "com" "^X" "user"
         "<" "@" "sun" "." "KCI" ">"
```

rule set 3   input:

```
"^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "@" "sun" "." "KCI"
```

rule set 8   input:

```
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "@" "sun" "." "KCI"
```

rule set 8 returns:

```
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "@" "sun" "." "KCI"
```

rule set 6   input:

```
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "<" "@" "sun" "." "KCI" ">"
```

rule set 6 returns:

```
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "<" "@" "sun" "." "KCI" ">"
```

rule set 3 returns:

```
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "<" "@" "sun" "." "KCI" ">"
```

rule set 24 returns:

```
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "%" "sun" "." "KCI"
                     "<" "@" "kciwrl" "." "kci" "." "com" ">"
```

Finally, rule set 4 is applied.

```
rewrite: rule set  4   input:
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
           "^X" "user" "%" "sun" "." "KCI"
               "<" "@" "kciwrl" "." "kci" "." "com" ">"

rewrite: rule set  4 returns:
        "^V" "tcp" "^W" "kciwrl" "." "kci" "." "com"
                "^X" "user" "%" "sun" "." "KCI" "@"
                        "kciwrl" "." "kci" "." "com"
```

If a TRANSFER mail user addresses the message to the gateway correspondent as

```
SMTPGATE(user@sun.kci)
```

the final address resolution of the suffix part user@sun.kci is

```
mailer    =     tcp
host      =     kciwrl.kci.com
recipient =     user%sun.KCI@kciwrl.kci.com
```

The SMTP gateway establishes a connection to host kciwrl.kci.com and sends the message to recipient user%sun.KCI@kciwrl.kci.com.

Configuring the OSS Environment to Use NonStop TCP/IP

This section describes considerations for accessing NonStop TCP/IP from Open Systems Services (OSS) sockets applications.

OSS Sockets Support Files

OSS sockets provide functions which have dependencies on files that have the following names in the /etc directory:

- hosts
- resolv.conf (Guardian name is RESCONF)
- networks
- protocols (Guardian name is PROTOCOL)
- services

Note. The /etc directory in the OSS environment is the equivalent to the ZTCPIP subvolume in the Guardian environment.

Note. You may need to be logged on as SUPER.SUPER to write to the /etc directory.

To establish these files in the /etc directory, you can either copy their equivalents from the Guardian subvolume ZTCPIP or establish a symbolic link to the files so that any changes made to the Guardian files also apply to OSS.

To set up a symbolic link in the OSS /etc directory to the Guardian file equivalents, enter the following commands at the OSS shell prompt:

```
cd /etc
ln -s /G/<volume>/ztcpip/resconf resolv.conf
ln -s /G/<volume>/ztcpip/networks networks
ln -s /G/<volume>/ztcpip/protocol protocols
ln -s /G/<volume>/ztcpip/services services
ln -s /G/<volume>/ztcpip/hosts hosts
```

Note. The volume varies; for example, the volume where files reside when you enter these commands may be <ISV volume> (Installation Subvolume) for INSTALL-based software distribution or <TSV volume> (Target Subvolume) for DSM/SCM-based software distribution or <SYSTEM> if a system administrator has moved them since software installation.

OSS Socket Behavior

Use the OSS shell DEFINE `TCPIP^PROCESS^NAME` to specify the name of the TCP/IP process to be used by the OSS socket application. For example, prior to running the socket application, enter the commands shown in Example 3-21 (substituting $ZTC*n* where *n* specifies the TCP/IP process of your choice).

Example 3-21. Setting the OSS DEFINE to Specify the TCP/IP Process

```
<shell prompt> del_define TCPIP^PROCESS^NAME
<shell prompt> add_define TCPIP^PROCESS^NAME file=\$ztc1
```

Syntax Considerations

Example 3-21 observes the following syntax rules:

- `TCPIP^PROCESS^NAME` should be in uppercase letters.
- The shell commands (for example, `del-define` and `add_define`) should be in lowercase letters.
- A backslash (\) should be included before the name of the TCP/IP process is specified. Otherwise, the OSS shell will not accept the dollar sign ($).

Note. You can also issue a Guardian DEFINE to specify the name of the TCP/IP process to be used by the OSS socket application. However, you must issue this DEFINE before invoking OSS. The TCP/IP process name can also be set programmatically (see the *OSS System Calls Reference Manual*).

Installing inetd

In the OSS environment, the inetd daemon provides services similar to and in extension to services provided by the LISTNER process in the Guardian environment.

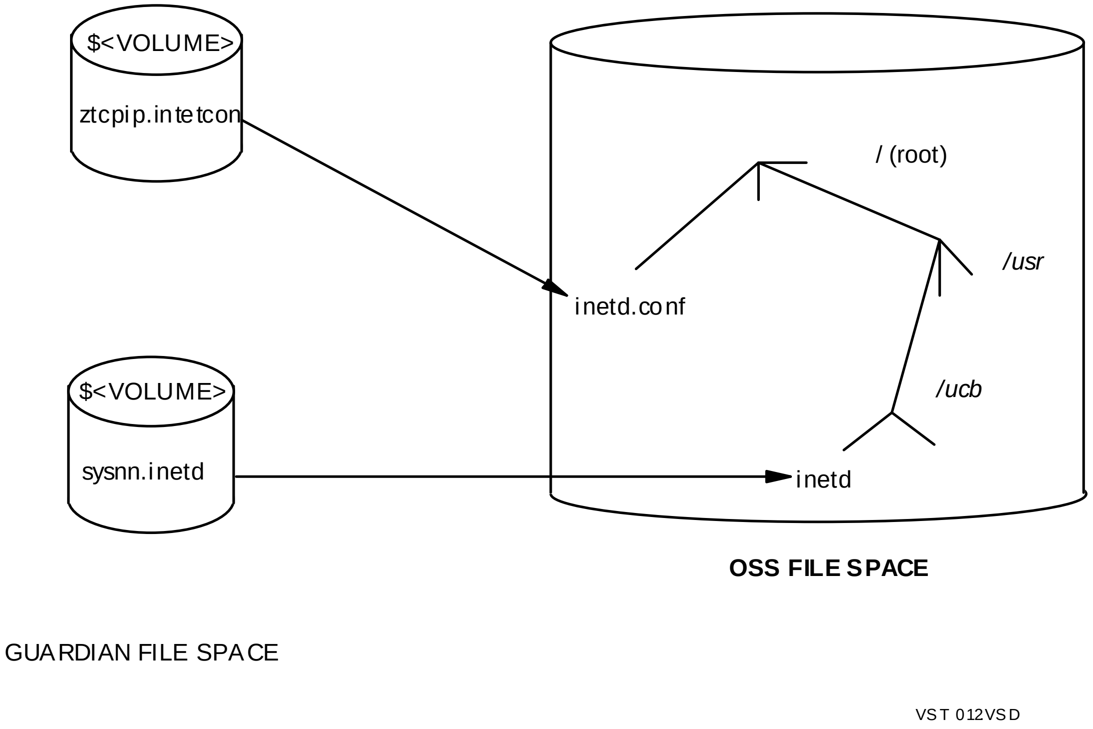
Installing inetd requires that its object file and corresponding configuration file be copied from a Guardian subvolume to OSS directories. To install inetd in the /usr/ucb directory and the configuration file in the /etc directory (the recommended location for these files), enter the following:

```
< shell prompt> cp /G/< volume>/SYSnn/inetd /usr/ucb/inetd
< shell prompt> cp /G/< volume>/ZTCPIP/inetconf /etc/inetd.conf
```

Note. See the Note on page 3-57 regarding <volume>.

Figure 3-6 shows how the above commands install the files.

Figure 3-6. Installing inetd and inetd.conf



Note. For more information about the inetd.conf file, see the *Open System Services Shell and Utilities Reference Manual*. The inetd.conf file is also documented online on the inetd man page.*

* To access the man page enter <shell prompt> man inetd.

Starting inetd

To start inetd, enter the following command at the shell prompt:

```
<shell prompt> /usr/ucb/inetd
```

If inetd is already running, you need to kill the first one before starting a new one. First, get the OSS process identification (OSS PID) of the running inetd daemon by using the ps command. Then, enter one of the following commands:

```
<shell prompt> kill -9 <OSS PID>
```

```
<shell prompt> kill -s KILL <OSS PID>
```

Upon starting, inetd reads in the inetd.conf file, located in the /etc directory.

Note. The inetd sockets application specifies a check for an environment variable to determine which TCP/IP process (Transport Provider) to use.

Preventing Port Collisions

Collisions between the services provided by the Guardian LISTNER process and the inetd daemon can occur if the LISTNER process in the Guardian environment and the inetd daemon in the OSS environment are assigned to the same TCP/IP process using the same port and protocol.

In the above case, the inetd daemon does not connect for a specific service, and an EMS event is generated. The inetd daemon retries the connection; EMS events accompany all subsequent rejections. To check for this condition, use the event viewer provided by TSM (G-series RVUs only). To prevent this condition, specify a TCP/IP process (Transport Provider) that doesn’t have LISTNER running on it by using the command shown in Example 3-21 or modify the configuration files to prevent port collisions as described below.

Modify the Configuration Files to Prevent Port Collisions

As an alternative to specifying a TCP/IP process other than the default, (and one that doesn’t have a LISTNER process), you can modify either one or both of the configuration files (portconf and inetd.conf). First, determine which services between the Guardian LISTNER process and the OSS inetd process conflict by looking at the portconf and inetd.conf files). Then, make one of the following modifications:

- Disable the conflicting services in the Guardian LISTNER’s portconf file (by commenting them out). In this case, the inetd daemon provides the service.
- Disable the conflicting services in the OSS inetd’s inetd.conf file (by commenting them out). In this case, the LISTNER provides the service.

- Split the service between the LISTNER process and the inetd daemon, (taking advantage of the fact that LISTNER supports only TCP ports), by commenting out the TCP port for the service in the inetd.conf file. In this case, the LISTNER process provides the service (on the TCP port) from the Guardian environment and the inetd daemon provides the service (on the UDP port) from the OSS environment.

Example 3-22 shows part of an inetd.conf file; it shows how to comment out a conflicting service between inetd and the Guardian LISTNER on the same TCP/IP process. This example shows the third option listed above (splitting the service, in this case echo, between LISTNER and inetd by commenting out the tcp port in the inetd.conf file.)

Example 3-22. Preventing Port Collisions by Modifying inet.conf

```
#echo      stream   tcp   nowait   super.super   internal
echo       dgram    udp   wait     super.super   internal
discard    stream   tcp   nowait   super.super   internal
discard    dgram    udp   wait     super.super   internal
daytime    stream   tcp   nowait   super.super   internal
daytime    dgram    udp   wait     super.super   internal
chargen    stream   tcp   nowait   super.super   internal
chargen    dgram    udp   wait     super.super   internal
```

Note. Whereas inetd supports both TCP and UDP, LISTNER just supports TCP.

Note. Other services besides echo may collide; apply the same commenting procedure for those services.

For more information about the inetd.conf file, see the Miscellaneous Files section in the *Open System Services Shell and Utilities Reference Manual.* The inetd.conf file is also documented online on the inetd man page. To access the man page, enter <shell prompt> man inetd.

4 SCF Reference

This section provides information about:

- The Subsystem Control Facility (SCF)
- SCF commands available for TCP/IP (to find a command quickly, see Table 4-5 on page 4-9.)
- The PTrace facility

SCF for NonStop TCP/IP

SCF provides an operator interface to an intermediate process, the Subsystem Control Point (SCP), which in turn provides the interface to the I/O processes of the various subsystems.

The QIO subsystem improves the I/O performance of LAN clients by providing a shared data segment and associated functions to manage the various resources. The QIO Manager handles all SPI requests to the subsystem.

The NonStop TCP/IP subsystem runs as a NonStop process pair on the NonStop operating system and supports subnets using Ethernet or token ring LANs and ATM or X25 networks.

- The Ethernet and SNAP SUBNETs use the SLSA subsystem to provide access to Ethernet local area networks.
- The ATM SUBNET uses the ATM subsystem to access the ATM network.
- The NonStop TCP/IP subsystem is a client of the SLSA subsystem.
- The X.25 SUBNET uses the X.25 Access Method (X25AM) subsystem for access to the Defense Data Network (DDN), X.25-based public data networks (PDNs), and other X.25-based networks. This X.25 interface is used to create virtual circuits to remote hosts, so that IP datagrams can be sent over the virtual circuits.

Object Types

You can monitor and control the NonStop TCP/IP subsystem by issuing commands that act on one or more NonStop TCP/IP subsystem objects. Each object has an object type and an object name. The object type describes the type of object. The object name uniquely identifies the object within the system.

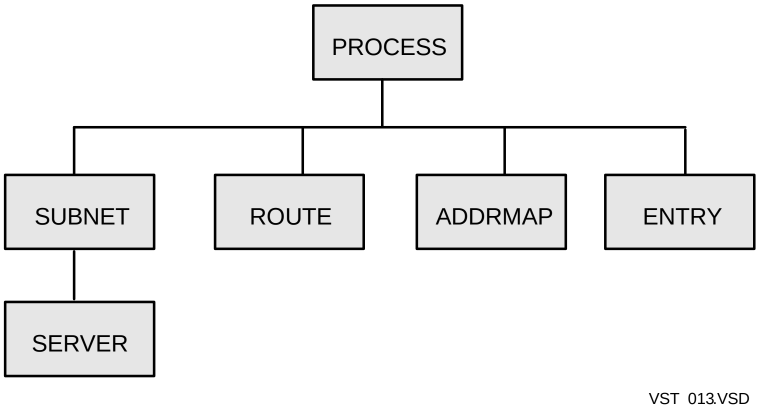
There are seven NonStop TCP/IP subsystem object types:

- ADDRMAP
- ENTRY
- null

- PROCESS
- ROUTE
- SERVER
- SUBNET

Figure 4-1 shows that the ROUTE, SUBNET, ENTRY, and ADDRMAP object types are peers. The ROUTE, SUBNET, ENTRY, and ADDRMAP object types are subordinate to the PROCESS object. The SERVER is subordinate to the SUBNET (and can only be added or deleted when the SUBNET is in the STOPPED state.) This hierarchy is important when issuing commands to the NonStop TCP/IP subsystem for processing. For example, because the ROUTE, SUBNET, ENTRY, SERVER and ADDRMAP object types are subordinate to the PROCESS object type, any commands pertaining to a ROUTE, SUBNET, ENTRY, SERVER or ADDRMAP object type can be issued only when the PROCESS object is in the STARTED summary state.

Figure 4-1. NonStop TCP/IP Subsystem Object Hierarchy



ADDRMAP Object Type

The ADDRMAP object specifies the name for the entry in the TCP/IP X25 address table (an internal table that maps IP addresses to X.121 addresses). The name must be preceded by a pound sign (#) and can have at most seven alphanumeric characters. The first character following the pound sign must be a letter.

You must assign a unique addr-name to each ADDRMAP object associated with a particular process. Names beginning with #ZADR are reserved for use by the process when creating dynamic entries. Use other letters to begin an address map name; for example, #ZADD.

ENTRY Object Type

The ENTRY object allows you to view and add to the Address Resolution Protocol (ARP and ATMARP) tables and obtain physical (MAC or ATM) addresses for any given IP address.

The ENTRY object also allows you to view the ATM address table which shows the current ATM, SVC, and PVC connections on the TCP/IP process. Each ATM address entry is displayed with the name set to ATMentry. The ATM address entries are used to track existing ATM connections. You cannot add or delete these type of entries with SCF.

An entry object is further broken down into one of the following types:

ARP	The entry is a standard ARP table entry where an IP address is mapped into a hardware MAC address.
ATMARP	The entry is an ATM ARP table entry.
ATM	The entry is an ATM address table entry.

The ENTRY object name must be preceded by a pound sign (#) and can have at most seven alphanumeric characters. The first character following the pound sign must be a letter. If the ENTRY object was added dynamically, the name is left blank.

null Object Type

The *`null`* object is not an actual object type. The term “null” is used to represent the lack of a specified object. Any SCF command that supports the *`null`* object type is issued without the specification of an object type. Commands support the *`null`* object type if an object type is irrelevant (as is the case with the VERSION command), or if they refer to a collection of objects (as is the case with the NAMES command).

To issue an SCF command using the *`null`* object, specify the name of the SCF command followed by a process name. The process name must be a valid process name. Do not use the term “null” when you issue the command.

PROCESS Object Type

The PROCESS object is the NonStop TCP/IP subsystem run as a process on the NonStop operating system. The PROCESS object automatically enters the started state when the NonStop TCP/IP program is invoked. A START PROCESS command is not necessary.

When you assign a name to a PROCESS object, you must make sure it conforms to the conventions for process names. The recommended form for TCP/IP process names is $ZTC*x* or $ZTC*xx*, where *x* is a letter or a numeric digit; for example, $ZTC0. Most of the NonStop client and server programs expect the name of a TCP/IP process to take this form. This convention allows applications to use a simple screening algorithm to locate NonStop TCP/IP processes in a system.

Note. In the NonStop TCP/IP subsystem, a TCP/IP process can have more than one IP address associated with it. However, each process must have a valid process name, and each IP address must be unique within the network.

ROUTE Object Type

The ROUTE object establishes the path a data packet travels to reach its destination. Instead of specifying a full path, a route specifies the packet's first host address and the packet's destination. The first host then routes the packet to the next appropriate address in-route to the destination. This sequence repeats until the packet reaches the destination.

Often, a NonStop system routes all packets to a default host, which in turn maintains a more complete routing table.

Each time you add a SUBNET, a route is created automatically. You can add more routes as necessary. Refer to Appendix B, NonStop TCP/IP Processes and Protocols, for a full explanation of routes and routing.

You must assign a unique ROUTE object name to each route associated with a given process. Precede the name with a pound sign (#). The ROUTE object name can have at most seven alphanumeric characters. The first character following the pound sign must be a letter. Table 4-1 shows an example of ROUTE object naming conventions.

Table 4-1. Route Object Naming Conventions

PROCESS $A	**ROUTE Object Name**
Route 1	#ABC1
Route 2	#ABC2
Route3	#ABC3

Routes that are associated with different processes can have identical names; therefore, you must specify the process name, a period, and the route name to distinguish a particular route on a particular process from other routes that might have the same name on other processes. To omit the process name and period and just specify the route name, set a default process name with the ASSUME command.

Names beginning with #RT*n* and #DRT*n* are reserved. Otherwise, use other letters to begin a route name; for example, #ROU5.

For further information on the ASSUME command, including the required syntax, refer to the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs*.

SERVER Object Type

The SERVER object allows you to add to a specific SUBNET the ATM address of ATMARP servers for use in resolving IP addresses. More than one server object can be added to a SUBNET but only one can be in use at a time. If communication is lost to an ATMARP server and a SUBNET is configured with more then one SERVER object, the next one in the list is tried.

The SERVER object name must be preceded by a pound sign (#) and can have at most seven alphanumeric characters. The first character following the pound sign must be a letter.

SUBNET Object Type

When you assign a name to a SUBNET object, you must assign a unique *subnet-name* to each SUBNET associated with a given process. The name must be preceded by a pound sign (#) and can have at most seven alphanumeric characters. The first character following the pound sign must be a letter.

Note. In this manual, when the word SUBNET is used to refer to the NonStop TCP/IP subsystem SCF object, it is capitalized. When subnet refers to the more general networking meaning (see Subnet Addresses on page B-5), it is not capitalized.

SUBNETs associated with different processes can have the same names; therefore, you must specify the process name, a period, and the SUBNET name in order to distinguish a particular SUBNET on a particular process from other subnets that might have the same name on other processes. To omit the process name and period, use the ASSUME command to set the default process. In subsequent commands, just specify the SUBNET name as in the following example:

```
1-> ASSUME PROCESS $ZTC0
2-> INFO SUBNET #SN1
```

HP recommends that you use the letters “SN” followed by a SUBNET number to identify a SUBNET; for example, $ZTC0.#SN1.

The SUBNET object is the point of connection between the TCP/IP process and the SLSA filter, ATM process, or X.25 I/O process (X25AM). Data coming from or going to an Ethernet LAN, IEEE 802.3 LAN, token-ring network, ATM network, or X.25 network goes through a SUBNET. Table 4-2 shows the SUBNETs and related devices.

Table 4-2. SUBNETs and Device Types for TCP/IP Processes

SUBNET Type	Device Type
ATM	ATM adapter
Ethernet	Ethernet adapters
SNAP	Token ring or Ethernet adapters
X.25	X25AM I/O process
loopback driver	

The name #LOOP0 is reserved for the loopback SUBNET.

Naming Convention Summary

Table 4-3 summarizes the reserved names for each object type and the naming convention rules.

Table 4-3. Object Naming Convention Summary and Reserved Names

Object Type	Reserved Names	Starting Symbol (Required)	First Character Requirement	First Character Recommendation	Character Limit
ADDRMAP	#ZADR	#	Letter	None	7
ENTRY	None	#	Letter	None	7
null	N/A	N/A	N/A	N/A	N/A
PROCESS	None	$	Letter	ZTC*x* where *x* is a letter or a numeric digit.	7
ROUTE	#RT*n* and DRT*n*	#	Letter	None	7
SERVER	None	#	Letter	None	7
SUBNET	#LOOP0	#	Letter	SN followed by a SUBNET number	7

Wildcard Support

Normally, an SCF command line must include an object specifier composed of the object type and an object name. For many commands, the NonStop TCP/IP subsystem accepts object-name templates. With an object-name template, one object name can be used to indicate that multiple objects of a given object type are to be affected by the command.

Object-name templates allow you to specify multiple objects by entering either a single wildcard character, or text and one or more wildcard characters. With the NonStop TCP/IP subsystem, you can use the following wildcard characters:

* Use an asterisk (*) to represent a character string of undefined length. The following example deletes all SUBNETs and routes subordinate to $ZTC0:

```
-> DELETE SUBNET $ZTC0.*
```

The following example deletes all SUBNETs subordinate to $ZTC0 that have names that start with #SN:

```
-> DELETE SUBNET $ZTC0.#SN*
```

The following example deletes all routes subordinate to $ZTC0 that start with #R and end with 5:

```
-> DELETE ROUTE $ZTC0.#R*5
```

? Use the question mark to represent a single unknown character in a specific position. For example, $ZTC0.#S?1 selects all object names subordinate to $ZTC0 that begin with #S, end with 1, and contain exactly one character between the #S and the 1.

You can use wildcard characters in any combination.

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 objects of the specified object type under the assumed process. For example, the next two commands set the default process to $ZTC1 and display information about all SUBNETs under $ZTC1:

```
1-> ASSUME PROCESS $ZTC1
2-> INFO SUBNET *
```

You cannot use object-name templates to specify the PROCESS object.

Summary States

The NonStop TCP/IP subsystem objects have operational states, known as summary states. The summary state of an object at a given instant is important; certain commands have no effect on an object when it is in one state but can affect the object when it is in another state.

The summary states supported by the NonStop TCP/IP subsystem are STARTED, STARTING, STOPPED, FREE, CONNECTING, REGISTERING, and REGISTERED. Table 4-4 shows the valid states for each object.

Table 4-4. Valid Object Summary States

Object	**STOPPED**	**STARTED**	**STARTING**	**FREE**	**CONNECTING**	**REGISTERING**	**REGISTERED**
ADDRMAP		X					
ENTRY		X					
null							
PROCESS		X					

Table 4-4. Valid Object Summary States

Object	STOPPED	STARTED	STARTING	FREE	CONNECTING	REGISTERING	REGISTERED
ROUTE	X	X					
SERVER				X	X	X	X
SUBNET	X	X	X				

- In the STARTED summary state, the object is available for data transfer.
- In the STOPPED summary state, the object is defined (that is, the object exists) but it is not available for data transfer. The STOPPED summary state is not applicable to the PROCESS object. If the PROCESS object is not STARTED, it is undefined (that is, the process does not exist).
- In the STARTING summary state, the object has been initialized and is attempting to start.
- In the FREE summary state, the server is not in use.
- In the CONNECTING summary state, the server is waiting for an ATM API Connect command to complete.
- In the REGISTERING summary state, the server is waiting for an ATMARP reply to an ATMARP request to register the SUBNET’s IP address with the ATMARP server.
- In the REGISTERED summary state, the SUBNET has successfully registered with the ATMARP server.

NonStop TCP/IP SCF Commands

This subsection contains the following information:

- A table describing the Subsystem Control Facility (SCF) commands supported by the NonStop TCP/IP subsystem and the object types supported for each command.
- Detailed information on object specification syntax.
- The following detailed information on each SCF command:
  - A description of the command function.
  - The command syntax.

- The object specification, which shows the supported object types and object names.
- Descriptions, by object type, of the attribute.
- Considerations you should be aware of before using the command.
- Command examples.

Note. The terms CPU and processor are used interchangeably in this section.

Supported Commands and Object Types

This section describes the SCF commands that are interpreted specifically for the NonStop TCP/IP subsystem. The *SCF Reference Manual for G-Series RVUs* and the *SCF Reference Manual for H-Series RVUs* provide general information about SCF commands. You should be familiar with that information before reading the NonStop TCP/IP subsystem-specific information provided here.

Table 4-5 lists the SCF commands and object types supported by the NonStop TCP/IP subsystem; (the page number of the command description follows the command name, and a page number under the object indicates both that the command is support for that object and the page number for that specific command).

Note. Note that Table 4-5 does not describe commands supported by the X25AM and SLSA subsystems. Such commands are covered in the SCF reference manuals for those subsystems. The titles of those reference manuals are listed in the About This Manual section.

Table 4-5. Commands and Object Types for NonStop TCP/IP SCF (page 1 of 2)

	Object Types						
SCF Command	ADDRMAP	ENTRY	PROCESS	ROUTE	SERVER	SUBNET	*null*
ABORT Command on page 4-12			4-12	4-13		4-13	
ADD Command on page 4-14	4-15	4-16		4-18	4-20	4-21	
ALTER Command on page 4-25	4-25		4-26			4-31	
DELETE Command on page 4-33	4-33	4-34		4-35	4-35	4-36	
INFO Command on page 4-37	4-37	4-38	4-41	4-45	4-47	4-48	
LISTOPENS Command on page 4-54			4-54				

Table 4-5. Commands and Object Types for NonStop TCP/IP SCF (page 2 of 2)

	Object Types						
SCF Command	**ADDRMAP**	**ENTRY**	**PROCESS**	**ROUTE**	**SERVER**	**SUBNET**	***null***
NAMES Command on page 4-58	4-58			4-59		4-60	
PRIMARY Command on page 4-61			4-61				
START Command on page 4-62				4-62		4-63	
STATS Command on page 4-64	4-64		4-65	4-85		4-87	
STATUS Command on page 4-90		4-90	4-93	4-99	4-100	4-102	
STOP Command on page 4-103			4-103	4-104		4-105	
TRACE Command on page 4-106			4-106			4-108	
VERSION Command on page 4-110			4-110				4-110

Table 4-6 lists the sensitive and nonsensitive NonStop TCP/IP SCF commands.

Table 4-6. Sensitive and Nonsensitive SCF Commands

Sensitive Commands	Nonsensitive Commands
ABORT	INFO
ADD	LISTOPENS
ALTER	NAMES
DELETE	STATS (without the RESET option)
PRIMARY	STATUS
START	VERSION
STATS (with the RESET option)	
STOP	
TRACE	

Entering SCF Commands

You start SCF interactively with the following TACL command:

```
1>SCF
```

It is seldom necessary to specify SCF RUN parameters because the default values are appropriate for most situations. For a more detailed description of the TACL RUN command parameters that apply to SCF, refer to the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs*.

At the beginning of an SCF session, SCF displays its product banner, which includes the product name, product number, version number, RVU date, and copyright statement.

SCF waits for a command, followed by a carriage return. After the command has been received and processed, SCF displays its prompt for the next command.

An SCF command always begins with a keyword identifying the command (such as ADD, ABORT, or ALTER).

The keyword is followed by the object specifier, consisting of the object type and the object name, as in the following example:

```
1-> ABORT SUBNET $ZTC0.#SN1
```

If additional attribute specifiers are required to define characteristics of the object, the object name is followed by a comma and the attribute name and value, as in the following example:

```
1-> ALTER PROCESS $ZTC0, DELAYACKS OFF
```

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

You can enter multiple SCF commands at a single prompt by separating the commands with semicolons, as in the following example:

```
1-> ASSUME PROCESS $ZTC0;ALTER, HOSTNAME "slugo1"
```

When processing a command line that contains more than one command, SCF executes the commands one at a time from left to right. If an error occurs, SCF displays the appropriate error message and ignores the rest of the line.

You can also continue a command that starts on one line onto a second line by terminating the first line with an ampersand (&). SCF prompts for additional input before executing the command, as in the following example:

```
1-> ADD SUBNET $ZTC0.#SN1, TYPE ETHERNET, &
1-> DEVICENAME LAN01, IPADDRESS 120.0.0.1
```

You must not enter more than 2048 characters for any input command.

Note. SCF accepts input from either a terminal or a disk (OBEY) file and directs output to either a terminal, disk file, or printer. However, in this manual, all examples assume that a terminal is being used for both input and output.

The rest of this section describes each SCF command for the NonStop TCP/IP subsystem.

ABORT Command

The ABORT command terminates the operation of specified NonStop TCP/IP subsystem processes, SUBNETs, or routes as quickly as possible. Only enough processing is done to ensure the integrity of the subsystem. The objects are left in the STOPPED summary state.

If there are any outstanding socket requests from the application, the ABORT command must be used instead of the STOP command. All pending requests to the I/O process(es) are canceled. In addition, all pending socket requests are canceled. The NonStop TCP/IP subsystem returns the error ECONNABORT to the application. If the application attempts to initiate a socket request, FS error 201 is returned.

This is a sensitive command.

ABORT PROCESS Command

The ABORT PROCESS command terminates the operation of a process as quickly as possible. This command stops and deletes the process object and all associated SUBNETs and routes.

Command Syntax

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

`OUT` *`file-spec`*

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

`PROCESS` *`process-name`*

is a valid process name indicating the desired NonStop TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command aborts and deletes a process named $ZTC0:

```
1-> ABORT PROCESS $ZTC0
```

ABORT ROUTE Command

The ABORT ROUTE command terminates the operation of a route as quickly as possible; only enough processing is done to ensure the integrity of the subsystem. The object is left in the STOPPED summary state.

Command Syntax

```
ABORT [ / OUT file-spec / ] [ ROUTE route-spec ]
```

`OUT` *`file-spec`*

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

`ROUTE` *`route-spec`*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *`route-spec`* has the form:

`$`*`process-name`*`.#`*`route-name`*

If you specify the SCF object type (ROUTE) or any portion of the object name (*`route-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command aborts a route named $ZTC0.#ROU1:

```
1-> ABORT ROUTE $ZTC0.#ROU1
```

ABORT SUBNET Command

The ABORT SUBNET command terminates the operation of a SUBNET as quickly as possible; only enough processing is done to ensure the integrity of the subsystem. The object is left in the STOPPED summary state.

Command Syntax

```
ABORT [ / OUT file-spec / ] [ SUBNET subnet-spec ]
```

OUT *file-spec*

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

SUBNET *subnet-spec*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *subnet-spec* has the form:

$*process-name*.#*subnet-name*

If you specify the object type (SUBNET) or any portion of the object name (*subnet-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Examples

The following command aborts a SUBNET named $ZTC0.#SN1:

```
1-> ABORT SUBNET $ZTC0.#SN1
```

The following command aborts all SUBNETs for the NonStop TCP/IP process:

```
1-> ASSUME PROCESS $ZTC0
2-> ABORT SUBNET *
```

Considerations

- The object-name template (wildcard notation) is supported.
- All activities being performed by the specified objects are halted. Files may be left in an inconsistent or incomplete state.
- Use the STOP command if you want to stop the operation of objects in a more controlled manner. The STOP command does not abruptly terminate activities in progress.

ADD Command

The ADD command adds a SUBNET or route to the NonStop TCP/IP subsystem. You must enter an ADD SUBNET command for each subnet with which the NonStop TCP/IP subsystem is to communicate. Each SUBNET defines a point of attachment through which data is sent or received.

This is a sensitive command.

ADD ADDRMAP Command

The ADD ADDRMAP command defines the name for the entry in the NonStop TCP/IP X.25 address table (an internal table that maps IP addresses to X.121 addresses).

An ADD ADDRMAP command must be entered for each X.121 address.

Command Syntax

```
ADD [ / OUT file-spec / ] [ ADDRMAP addrmap-spec ]

   , IPADDRESS ip-address
   , X121ADDR x25-address
```

`OUT` *`file-spec`*

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

`ADDRMAP` *`addrmap-spec`*

defines the name to be used for the entry in the NonStop TCP/IP X.25 address table. The fully-qualified *`addrmap-spec`* has the form:

`$`*`process-name`*`.#`*`addrmap-name`*

If you specify the SCF object type (ADDRMAP) or any portion of the object name (*`addrmap-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`IPADDRESS` *`ip-address`*

specifies the Internet address for the *`addrmap-name`* entry in dotted decimal format. An Internet address is a 4-octet (32-bit) numeric value identifying a particular network (network address portion), and a local host on that network (local address portion) as defined in RFC 1010.

Default: None.

`X121ADDR` *`x25-address`*

specifies the X.25 address for the *`addrmap-name`* entry.

Default: None.

Example

```
1-> ADD ADDRMAP $ZTC0.#A1,IPADDRESS 123.456.78.9,&
1-> X121ADDR "1234567890123456"
```

ADD ENTRY Command

The ADD ENTRY command creates an entry in an ARP or ATMARP table. This is a sensitive command.

Command Syntax

```
ADD [ /OUT file-spec/ ] [ ENTRY entry-spec]
          , TYPE { ARP | ATMARP }
          , IPADDRESS  ip-addr
          [, MACADDR  mac-address ]
          [, ATMADDR atm-address ]
          [, PVCNAME “name” ]
          [, SUBNET “name” ]
```

OUT *file-spec*

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

ENTRY *entry-spec*

specifies the name of the ENTRY object. The fully-qualified *entry-spec* has the form:

$*process-name*.#*entry-name*

If you specify the SCF object type (ENTRY) or any portion of the object name (*entry-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

ATMADDR *atm-address*

ATMADDR is the ATM address that is associated with this entry. Only NSAP formatted addresses are allowed and must be entered as 40 hex digits. The *atm address* portion must be enclosed in parentheses with only the following format accepted:

```
ATMADDR
(ISONSAP:%H1234567890123456789012345678901234567890)
```

The ATMADDR attribute is only valid for entries of type ATMARP.

TYPE ARP | ATMARP

specifies the ENTRY type. A TYPE of ARP maps an IP address with an Ethernet or token-ring MAC address. A TYPE of ATMARP maps an IP address with either a destination’s ATM address or the PVC name being used on the ATM adapter. Both types require an IP address. The TYPE ARP also requires the MACADDR attribute. The TYPE ATMARP requires the ATMADDR attribute for SVC connections. For PVC connections, the PVCNAME and SUBNET attributes, not the ATMADDR attribute, are required.

`IPADDRESS` *`ip-addr`*

specifies the internet address for the entry and is specified in dotted decimal notation.

`MACADDR` *`mac-address`*

specifies the Ethernet or token-ring address for the ENTRY. It is entered as a string of twelve hexadecimal digits preceded by a “%”. The MACADDR attribute is only valid for entries of type ARP.

`PVCNAME` *`name`*

PVCNAME is the name of the PVC on the ATM adapter being used. The adapter name is not included. The PVCNAME attribute is only valid for entries of type ATMARP.

`SUBNET` *`name`*

SUBNET specifies the subnet on which a PVC should be associated. The SUBNET attribute is only valid for entries of type ATMARP and is required if PVCNAME is supplied.

Examples

The following commands create entries in the ARP or ATMARP entry table:

```
1-> ASSUME PROCESS $ZTC0
2-> ADD ENTRY #A1, TYPE ARP, IPADDRESS 1.2.3.4, &
2-> MACADDR %H08008E003578

3-> ADD ENTRY #A2, TYPE ATMARP, IPADDRESS 1.2.3.5, &
3-> ATMADDR (ISONSAP:%H1234567890123456789012345678901234567890)

4-> ADD ENTRY #A4, TYPE ATMARP, IPADDRESS 1.2.3.7, &
4-> PVCNAME “#IP.PVC000”, SUBNET “#SN1”
```

ADD ROUTE Command

The ADD ROUTE command creates a route.

Command Syntax

```
ADD [ / OUT file-spec / ] [ ROUTE route-spec]

     , DESTINATION destination-ip-address
     , GATEWAY     gateway-ip-address
   [ , DESTTYPE    { HOST | BROADCAST }   ]
   [ , NETMASK      mask-val   ]
   [ , METRIC       metric-val ]
```

`OUT` *`file-spec`*

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

`ROUTE` *`route-spec`*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *`route-spec`* has the form:

```
 $process-name.#route-name
```

If you specify the SCF object type (ROUTE) or any portion of the object name (*`route-spec`*) in prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`DESTINATION` *`destination-ip-address`*

specifies the Internet address of a single host or an entire network which can be reached through the system specified in GATEWAY. A zero in the local address portion of the destination Internet address acts as a wildcard, representing all hosts on the network specified in the network portion of the Internet address. If the destination Internet address is 0 (0.0.0.0), this route specifies the default router or gateway. All packets with addresses for which routes cannot be determined are sent to the host specified in GATEWAY.

This parameter is required.

Default: If the route is added automatically when a SUBNET is added, the default address is the IP address of the SUBNET's host (the value for the SUBNET's IPADDRESS attribute) converted to broadcast form.

GATEWAY *gateway-ip-address*

specifies the Internet address of the router or gateway host through which the network or host addressed in DESTINATION can be reached. This is a required attribute.

Default: The subsystem supplies no default value for it unless the route is being added automatically as a part of the ADD SUBNET operation. The default value is the IP address of the SUBNET's host (the value for the SUBNET's IPADDRESS attribute).

DESTTYPE

specifies whether the route is a connection to a specific host (HOST) or to a network (BROADCAST). This is an optional attribute.

Default: If you do not specify DESTTYPE in an ADD ROUTE command, the default value is BROADCAST. If a route is added automatically as the result of an ADD SUBNET command, the default value is BROADCAST.

NETMASK *mask-val*

specifies a subnet mask value to be associated with the route entry. This attribute allows NonStop TCP/IP to interface with routers that support and use Classless Inter-Domain routing (CIDR).

Specify `mask-val` in dotted-decimal or hexadecimal notation. If `mask-val` is not specified, it defaults to the default network mask of the specified IP address. The netmask value is set to 0.0.0.0 for the network routes to IP address of 0 (0.0.0.0). The `NETMASK` value is set to 255.255.255.255 for the host route. Because a non-contiguous mask value is not supported, netmask values such as 0.255.255.0, 255.0.255.0, 0.0.255.0 are invalid.

METRIC *metric-val*

indicates the number of hops to the destination. The `METRIC` attribute is optional for add commands; If you do not specify `METRIC,` its value defaults to

- Zero (0) if the destination is on a directly-attached network.
- One (1) if the route utilizes one or more routers or gateways. Note that this default value assumes that there is only one hop.

Examples

The following command adds a route, named #ROU1, to the process $ZTC0:

```
1-> ADD ROUTE $ZTC0.#ROU1, DESTINATION 120.0.1.5, &
1-> GATEWAY 120.0.1.0, DESTTYPE HOST
```

ADD SERVER Command

The ADD SERVER command adds to a SUBNET the ATM addresses of ATMARP servers to use when trying to resolve the ATM address associated with an IP address. The ADD SERVER command requires both the SUBNET and ATMADDR attributes. The SUBNET name is entered as a quoted ASCII character string. The SUBNET must be in the STOPPED state before the ADD SERVER command is issued. This is a sensitive command.

Command Syntax

```
 ADD [ /OUT file-spec/ ] [ SERVER server-spec ]
          , SUBNET “name”
          , ATMADDR atm address
```

`OUT` *`file-spec`*

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

`ATMADDR`

specifies the ATM address of the ARP server.

`SERVER` *`server-spec`*

specifies the name of the SERVER object. The fully-qualified *`server-spec`* has the form:

```
 $process-name.#server-name
```

If you specify the SCF object type (SERVER) or any portion of the object name (*`server-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`SUBNET` *`“name”`*

specifies the name of the SUBNET to which to add the server name.

Example

The following command adds an ATMARP server to use in resolving ATM addresses to the SUBNET named #EN1:

```
1-> ASSUME PROCESS $ZTC0
2-> ADD SERVER #SR1, SUBNET “#EN1”, &
2-> ATMADDR (ISONSAP:%H1234567890123456789012345678901234567890)
```

Considerations

- This command requires both the SUBNET and ATMADDR attributes.

- The SUBNET name is entered as a quoted ASCII character string.
- The SUBNET must be in the STOPPED state before ADD SERVER can be issued to the SUBNET.

ADD SUBNET Command

The ADD SUBNET command creates an SCF SUBNET object. The SUBNET object associates a PROCESS object with a LIF, thereby connecting the NonStop TCP/IP process to a port on a communications adapter or X25AM line.

Command Syntax

```
ADD [ / OUT file-spec / ]  [ SUBNET [ subnet-spec ]

   , TYPE { LOOPBACK | ATM | ETHERNET | SNAP | X25 }
   , DEVICENAME device-name
   , IPADDRESS  ip-address
   [ , SUNAME   subdevice-name ]
   [ , IRDP { ON | OFF }       ]
   [ , ARPSERVER { ON | OFF }  ]
   [ , ATMSEL byte-number      ]
   [ , GATEWAY ON|OFF          ]
   [ , SUBNETMASK mask-val     ]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device, such as an FESA, E4SA, GESA, G4SA, TRSA, or ATM3SA or an X25AM line. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`*`.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`TYPE LOOPBACK | ATM | ETHERNET | SNAP | X25`

specifies the type of SUBNET to be added. The type values are: LOOPBACK, ETHERNET, ATM, SNAP, and X25. This parameter is required.

Default: None.

`DEVICENAME` *`device-name`*

is the name of the SLSA LIF that provides access to the Ethernet or token-ring LAN, is the name of the ATM adapter, or is the name of the X25AM line that provides access to the X.25 network. For information on how to choose a SLSA device name, see the Hint on page 1-7. For information on how to choose the ATM device name, see the Hint on page 1-13.

When adding a SUBNET, the DEVICENAME for the SLSA SUBNET type does not begin with a dollar ($) character. X25 SUBNETs still require the dollar ($) character as do ATM SUBNETs.

Default: None.

`IPADDRESS` *`ip-address`*

specifies the Internet address of a host or the conventional loopback address, which is 127.0.0.1. You can use the loopback address for testing. For more information on IP addresses, refer to the Appendix B, NonStop TCP/IP Processes and Protocols. This parameter is required.

Default: None.

`SUNAME` *`subdevice-name`*

defines the subdevice name. This attribute is valid only for type X.25 SUBNETs. (Prior to the D30 RVU, the SUNAME had to be set using a TACL PARAM command before a SUBNET could be added using SCF. Beginning with the D30 RVU, you cannot use PARAM commands in the TACL environment to specify SUNAME for the NonStop TCP/IP subsystem—the subdevice name now is defined by using this attribute.)

`IRDP ON | OFF`

enables (ON) or disables (OFF) the ICMP Router Discovery Protocol on the SUBNET interface. IRDP is a mechanism for locating default routers and is specified in RFC 1256. IRDP also must be enabled on any local LAN routers. If redundant routers are configured with route hold-down times and advertisement intervals of approximately 30 seconds, IRDP can be used to provide a “black hole”, or dead gateway, detection mechanism. The NonStop TCP/IP subsystem implements IRDP using IP broadcasts rather than IP multicasts.

Default: OFF.

`ARPSERVER ON | OFF`

enables (ON) or disables (OFF) the SUBNET to act as an ATMARP server. The SUBNET must be of type ATM for this to function. The default is OFF.

`ATMSEL` *`byte-number`*

is the ATM selector byte to use on an ATM type SUBNET. The default is 0 and valid value s are 0 to 255.

`GATEWAY ON | OFF`

ON enables the SUBNET to add a default IP address filter to Ethernet LAN adapters. For the NonStop TCPIP process to operate as a Gateway, you must configure it with more than one SUBNET, and you must add each SUBNET configured on an Ethernet LAN adapter with this option enabled. Only one default IP filter can be added to each Ethernet LAN adapter. A SUBNET fails to enter the STARTED state if a SUBNET on another TCPIP process has already been added with the GATEWAY attribute enabled on the same device. The default is OFF.

`SUBNETMASK` *`mask-val`*

identifies which portion of the IP local address represents the subnet number and which part represents the host ID. If bits in the subnet mask are set to 1, the corresponding bits in the IP address are part of the network (and subnet) address. If bits in the subnet mask are set to 0, the corresponding bits in the IP address are part of the host ID (that is, that portion of the local address masked with 1s identifies the subnet, and the remainder of the local address uniquely identifies a host connected to the subnet). The *`mask-val`* is specified in dotted-decimal or hexadecimal notation. If it is not specified, *`ask-val`* defaults to the default network mask of the specified IP address. Non-contiguous mask values are not supported. Since non-contiguous mask values are not supported, netmask values such as 0.255.255.0, 255.0.255.0, 0.0.255.0 are considered invalid.

Examples

The first command in the following example, adds a new SUBNET named #SN1 to the PROCESS named $ZTC0. The SUBNET is of type ETHERNET and uses LAN01 as its point of connection to the Ethernet LAN. The SUBNET receives packets directed to IPADDRESS 120.0.0.1. This command also automatically adds the first route to this SUBNET, named #ROU1, giving it default values for the GATEWAY, DESTINATION, and DESTTYPE attributes:

```
1-> ADD SUBNET $ZTC0.#SN1, TYPE ETHERNET, &
1-> DEVICENAME LAN01, IPADDRESS 120.0.0.1
```

The following command adds a SUBNET of type ATM:

```
1-> ASSUME PROCESS $ZTC0
2-> ADD SUBNET #SN2, TYPE ATM, DEVICENAME $AM1, &
2-> IPADDRESS 172.16.192.200, ARPSERVER ON, ATMSEL 1
```

Considerations

- See Table 4-3 on page 4-6 for naming conventions and reserved object names.

- The NonStop TCP/IP X.25 interface can open a maximum of 128 subdevices on a specified X25AM device (line). The specified X25AM device must be active and the subdevices must be added and in the STARTED summary state before you add an X.25 SUBNET. Refer to the *X25AM Configuration and Management Manual* for information on the methods for adding X.25 subdevices.
- When you specify the name of the route or SUBNET you are adding, be sure to specify the process name in the ASSUME command or in the ADD command, as shown in the examples. Verify that the name is unique for that process.
- Do not add more than one route to the same network. No matter how many additional routes are added, only one route is used. Should that route become disconnected or should errors occur, you must stop the faulty route and start a new one.
- The SLSA subsystem must be operational for the ADD command to complete successfully. For more information, refer to the *LAN Configuration and Management Manual*.
- When a SUBNET is added, a corresponding route to this SUBNET is added automatically. Both the SUBNET and the route are placed in the STOPPED state. To initiate the operation of the object, you must start it with the START command.
- When adding a SLSA SUBNET type, the DEVICENAME does not begin with a “$” character. (X25 SUBNET types still require the “$” character.)
- The logical interface (LAN device) which you specify in the SCF ADD SUBNET command must be accessible from the TCP/IP processes’ primary processor (CPU) and potentially accessible from the TCP/IP processes’ backup processor (CPU). The LAN device also must be currently accessible from the same processor as the TCP/IP processes’ primary CPU. If both conditions are not met, the ADD of the SUBNET fails with the error: “Device access not available from same CPU pair as TCPIP.”
- SUBNETs on multiple TCP/IP processes in different or shared CPUs can share a LIF but only one SUBNET can own the IP default filter (have GATEWAY ON). For a TCP/IP process to be used as a router (in which case the process has multiple SUBNETs on different LIFs), each SUBNET must be added with the GATEWAY attribute enabled (ON).
- Although SUBNETs on multiple TCP/IP processes in different or shared CPUs can share a LIF, only one SUBNET from each process is allowed on the LIF.

ALTER Command

The ALTER command changes attribute values associated with the specified TCP/IP object. This is a sensitive command.

ALTER ADDRMAP Command

The ALTER ADDRMAP command is used to change the attribute values associated with a particular address map name.

Command Syntax

```
ALTER [ / OUT file-spec / ] [ ADDRMAP addrmap-spec]

        [ , IPADDRESS ip-address  ]
        [, X121ADDR x25-address   ]
```

`OUT` *`file-spec`*

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

`ADDRMAP` *`addrmap-spec`*

specifies the address map name to which the attribute changes apply. The fully-qualified *`addrmap-spec`* has the form:

`$`*`process-name`*`.#`*`addrmap-name`*

If you specify the SCF object type (ADDRMAP) or any portion of the object name (*`addrmap-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`IPADDRESS` *`ip-address`*

specifies the Internet address for the *`addrmap-name`* entry in dotted decimal format. An Internet address is a 4-octet (32-bit) numeric value identifying a particular network (network address portion), and a local host on that network (local address portion) as defined in RFC 1010.

`X121ADDR` *`x25-address`*

specifies the X.25 address for the *`addrmap-name`* entry.

Example

The following command changes the X121ADDR attribute of the specified ADDRMAP:

```
1->ALTER ADDRMAP $ZTC0.#A2, X121ADDR "0987654321654321"
```

ALTER PROCESS Command

The ALTER PROCESS command is used to change the attribute values of a process.

Command Syntax

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

   [   , TCPSENDSPACE            window-size           ]
   [   , TCPRECVSPACE            window-size           ]
   [   , UDPSENDSPACE            window-size           ]
   [   , UDPRECVSPACE            window-size           ]
   [   , DELAYACKS               { ON | OFF }          ]
   [   , DELAYACKSTIME           delayacks-time        ]
   [   , HOSTNAME               "host-name"            ]
   [   , HOSTID                  host-id               ]
   [   , TCPKEEPIDLE             keepidle-time         ]
   [   , TCPKEEPCNT              keepalive-retry-count ]
   [   , TCPKEEPINTVL            keepalive-retry-time  ]
   [   , DEBUG                   { ON | OFF }          ]
   [   , FULLDUMP                { ON | OFF }          ]
   [   , ALLNETSARELOCAL         { ON | OFF }          ]
   [   , TPCOMPAT42              { ON | OFF }          ]
   [   , TCPPATHMTU              { ON | OFF }          ]
   [   , EXPANDSECURITY          { ON | OFF }          ]
   [   , TCPTIMEWAIT             { ON | OFF }          ]
   [    ,ARPTIMER-REFRESHED      { ON | OFF }          ]
   [    ,RFC1323-ENABLE          { ON | OFF }          ]
   [    ,TCP-INIT-REXMIT-TIMEOUT int                 ]
   [    ,TCP-MIN-REXMIT-TIMEOUT  int                 ]
   [    ,TCP-LISTEN-QUE-MIN      int                 ]
   [    ,INITIAL-TTL             int                 ]
```

`OUT` *`file-spec`*

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

`PROCESS` *`process-name`*

is a valid NonStop process name of the TCP/IP process. If you omit the object type or object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`TCPSENDSPACE` *`window-size`*

specifies the default window size used to send data for the TCP protocol. You can specify values in the range 512 bytes through 262144 bytes.

Default: 8K bytes.

`TCPRECVSPACE` *`window-size`*

specifies the default window size used to receive data for the TCP protocol. You can specify values in the range 512 bytes through 262144 bytes. The value you specify affects NonStop TCP/IP subsystem performance. Although some applications may use values in the lower range, you should not set the value below 2K bytes.

Default: 8K bytes.

`UDPSENDSPACE` *`window-size`*

specifies the default window size used to send data for the UDP protocol. You can specify values in the range 512 bytes through 262144 bytes.

Default: 9216 bytes.

`UDPRECVSPACE` *`window-size`*

specifies the default window size used to receive data for the UDP protocol. You can specify values in the range 512 bytes through 262144 bytes. The value you specify affects the NonStop TCP/IP subsystem performance. Although some applications may use values in the lower range, you should not set the value below 2K bytes.

Default: 20800 bytes.

`DELAYACKS`

specifies whether the acknowledgment (ACK) should be delayed when a TCP packet is received from a remote site. The value ON delays acknowledgment for the duration of the interval specified by DELAYACKSTIME or until the TCP receive window is full, whichever comes first. Delaying acknowledgment allows multiple packets to be acknowledged with a single ACK, thus reducing network traffic. Depending on the attribute value for DELAYACKSTIME, it is possible for the TCP receive window to fill before an ACK is generated. Change it to OFF if you do not want the delay option to be in effect; that is, if you want an ACK sent after each packet is received.

Default: ON.

`DELAYACKSTIME` *`delayacks-time`*

specifies the length of delay before an ACK is sent for a packet when the DELAYACKS attribute value is ON. This attribute value is expressed in intervals of .01 seconds. The range is 0 through 50 (0 to .5 seconds).

Default: 5 (50 milliseconds).

`HOSTNAME` *`host-name`*

specifies the name by which the NonStop system running the TCP/IP process is known on the Internet. It is a character string of 49 or fewer characters, enclosed in quotation marks. This value only supports the gethostname() socket call.

Default: None.

`HOSTID` *`host-id`*

specifies the 32-bit number assigned to the NonStop system running the TCP/IP process. This is usually the host number portion of the IP address. This value is used only to support the gethostid socket call.

Default: 0D.

`TCPKEEPIDLE` *`keepidle-time`*

specifies the amount of time, in seconds, before TCP issues a keep-alive packet on sockets that have enabled this option. The value must be in the range 1 through 7200.

Default: 45 seconds.

`TCPKEEPCNT` *`keepalive-retry-count`*

specifies the number of times a keep-alive packet is sent without receiving an acknowledgment, after which the TCP connection is dropped. This value must be in the range 1 through 20.

Default: 8 times.

`TCPKEEPINTVL` *`keepalive-retry-time`*

specifies the time interval, in seconds, between retransmissions of unacknowledged keep-alive packets. The value must be in the range 1 through 1260.

Default: 45 seconds.

`DEBUG ON | OFF`

specifies whether or not additional TCP internal information is displayed. This attribute is used by HP support personnel for purposes of problem solving. ON declares that additional information is displayed. OFF disables the display of additional information.

Default: OFF.

`FULLDUMP ON | OFF`

specifies whether or not the QIO segment is saved when the TCP/IP primary process terminates abnormally (abends). Normally, to conserve disk space, only the TCP/IP process stack is saved. ON declares that the QIO segment is saved together with the TCP/IP process stack. OFF declares that only the TCP/IP process stack is saved.

Default: ON.

`ALLNETSARELOCAL ON | OFF`

The default is ON. ON causes TCP to use the interface MTU as a base for determining the TCP Maximum Segment Size (MSS) for each non-local TCP connection. A non-local TCP connection is one that goes to another network (not just another subnetwork).

If ALLNETSARELOCAL is OFF, TCP conforms to RFC-specified behavior and uses 512 bytes as the default MSS for non-local segments. For example, for Ethernet, when ALLNETSARELOCAL is ON, the non-local MSS is 1460; setting ALLNETSARELOCAL to ON can improve performance.

`TPCOMPAT42 ON | OFF`

is the flag used to set the TCP/IP process compatible with BSD4.2 versions as follows: if the flag is ON, then the original ACK minus 1 is sent in the keepalive packet; if the flag is OFF, the original ACK is sent in the keepalive packet.

Default The default value of this flag is ON.

`TCPPATHMTU ON | OFF`

is ON to cause TCP to use PATH MTU discovery on all TCP-type sockets (SOCK_STREAM) unless disabled on a socket by the SETSOCKOPT for SO_PMTU. The default for this option is OFF. PATH MTU is not supported on UDP sockets. The option, when enabled, only affects new TCP sockets; existing sockets are not changed.

`EXPANDSECURITY ON | OFF`

is ON to cause TCP to check if a SOCKET request from another Expand node has passed the Expand security check. This means the user is valid on this system and has correct remote passwords. If the check fails then the SOCKET request is rejected with file error 48. The default for this option is OFF

`TCPTIMEWAIT ON | OFF`

is the amount of time in seconds that a TCP connection remains in the TIME_WAIT state. The default is 60 seconds. The range is 1 to 120.

`ARPTIMER-REFRESHED ON | OFF`

is ON to cause TCP to restart the ARP timer every time the ARP table entry is referenced when transmitting an IP packet. The default for this option is OFF. If this option is enabled stale ARP table entries may take longer to be flushed.

`RFC1323-ENABLE ON | OFF`

is ON to cause TCP to support TCP Large Windows as documented in RFC 1323. When this option is enabled the TCP/IP process will use the TCP Window Scale and Timestamp options as described in RFC 1323. The largest TCP window supported is 262144 bytes when this option is enabled, and 65535 when the option is disabled. The default for this option is ON.

`TCP-INIT-REXMIT-TIMEOUT` *`int`*

is the initial retransmit timer value in milliseconds to use on a TCP connection. When the first round trip timer measurement is made on a TCP connection and the calculation is done to arrive at the retransmission timeout to use on the next packet sent, this value will be used unless the calculated value is larger. This variable can be used to help reduce the number of premature retransmission timeouts. The default is 1000 milliseconds, or 1 second. The range is 200 to 30000 milliseconds.

`TCP-MIN-REXMIT-TIMEOUT` *`int`*

is the minimum value allowed for the TCP retransmission timeout. If this value is too low the TCP/IP process may generate premature retransmissions. If this value is set too high then real retransmissions will be delayed increasing the time for error recovery. The default is 400 milliseconds. The range is 50 to 30000 milliseconds.

`TCP-LISTEN-QUE-MIN` *`int`*

is the minimum queue length that will be set on a TCP socket when the TCP/IP process handles a socket LISTEN or ACCEPT_NW1 function call. This value will be used if the queue length specified in the socket request is lower, otherwise the queue length in the socket request will be used. The default value is 5. The range is 1 to 1024.

`INITIAL-TTL` *`int`*

specifies the initial value for UDP and TCP TTL. The default value is 30, but you can use the ALTER command to specify 64.

Example

The following command turns off the delay of ACKs for the process $ZTC0:

```
1-> ALTER PROCESS $ZTC0, DELAYACKS OFF
```

The following command turns on PATH MTU discovery:

```
1-> ALTER PROCESS $ZTC0, TCPPATHMTU ON
```

ALTER SUBNET Command

The ALTER SUBNET command is used to change the attribute values of a SUBNET.

Command Syntax

```
ALTER [ / OUT file-spec / ] [ SUBNET subnet-spec ]

   [ [ , IPADDRESS   ip-address                         ]
     [ , SUBNETMASK  subnet-mask                        ]
     [ , IRDP { ON | OFF }                              ]
     [ , ADDALIAS ip-addr, SUBNETMASK %H0..FFFFFFFF ]
     [ , DELETEALIAS ip-addr                            ]
     [ , MTU size                                       ]
     [ , X25IDLETIME time                               ]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`*`.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`IPADDRESS` *`ip_address`*

is the 32-bit integer that identifies the subnet. This is the IP address assigned to the SUBNET.

`SUBNETMASK subnet-mask`

is a 32-bit integer that specifies the subnet mask to be used with this subnet. A subnet mask identifies which portion of the IP local address represents the subnet number and which part represents the host ID. If bits in the subnet mask are set to 1, the corresponding bits in the IP address are part of the network (and subnet) address. If bits in the subnet mask are set to 0, the corresponding bits in the IP address are part of the host ID (that is, that portion of the local address masked with 1s identifies the subnet, and the remainder of the local address uniquely identifies a host connected to the subnet). The *`mask-val`* is specified in hexadecimal notation. If it is not specified, *`ask-val`* defaults to the default network mask of the specified IP address. Non-contiguous mask values are not supported.

Since non-contiguous mask values are not supported, netmask values such as 0.255.255.0, 255.0.255.0, 0.0.255.0 are considered invalid.

`IRDP`

enables (ON) or disables (OFF) the ICMP Router Discovery Protocol on the SUBNET interface. IRDP is a mechanism for locating default routers and is specified in RFC 1256. IRDP must also be enabled on any local LAN routers. If redundant routers are configured with route hold-down times and advertisement intervals of approximately 30 seconds, IRDP can be used to provide a “black hole,” or dead gateway, detection mechanism. The NonStop TCP/IP subsystem implements IRDP using IP broadcasts rather than IP multicasts.

`ADDALIAS` *`ip-addr`*

allows the addition of the alias IP address to the SUBNET specified in the ALTER SUBNET command. The IP alias feature allows a process to be known to the Internet by different IP addresses. For a display example of an alias, see INFO SUBNET Display Format on page 4-49.

`DELETEALIAS`

allows the deletion of alias IP addresses that have been added by the ADDALIAS attribute.

`MTU` *`size`*

changes the interface MTU size on SNAP-type SUBNETs when configured on token-ring adapters. The minimum value is 512 bytes. The maximum value is 65526 bytes or the maximum supported by the token-ring LAN. The default for 4 mega-bit LANs is 4464 and for 16 mega-bit LANs is 17792.

`X25IDLETIME` *`time`*

changes the idle timeout value on X.25 type SUBNETs used to timeout X.25 connections. The default value is 2 minutes. The range accepted is 0 to 5000 minutes. If the time value is set to zero, the X.25 connections do not time out. Setting the time value to zero or to a value larger than the default may result in limiting the number of devices that can use the SUBNET. so be careful when you alter this value.

Example

The following command alters the subnet mask and IPADDRESS for the SUBNET $ZTC0.#SN1:

```
1-> ALTER SUBNET $ZTC0.#SN1, IPADDRESS 120.1.1.12, &
1-> SUBNETMASK %hFFFF0000
```

Considerations

- See Table 4-3 on page 4-6 for naming conventions and reserved object names.
- The object must be in the STOPPED summary state when the ALTER command is issued.
- When the ALTER command is completed, the object remains in the same summary state that existed before you issued the command.
- Use the INFO command to view the current attribute values.

DELETE Command

The DELETE command removes address map names, SUBNETs, and routes from the NonStop TCP/IP subsystem. You cannot delete a process.

This is a sensitive command.

DELETE ADDRMAP Command

The DELETE ADDRMAP command removes a specific address map name entry from the NonStop TCP/IP X.25 address table (an internal table that maps IP addresses to X.121 addresses).

Command Syntax

```
DELETE [ / OUT file-spec / ] [ ADDRMAP addrmap-spec]
```

`OUT` *`file-spec`*

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

`ADDRMAP` *`addrmap-spec`*

defines the name to be used for the entry in the NonStop TCP/IP X.25 address table. The fully-qualified *`addrmap-spec`* has the form:

```
$process-name.#addrmap-name
```

If you specify the SCF object type (ADDRMAP) or any portion of the object name (*`addrmap-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command deletes the specified ADDRMAP:

```
1->DELETE ADDRMAP $ZTC0.#A2
```

DELETE ENTRY Command

The DELETE ENTRY command removes entries from the ARP or ATMARP tables. Entries can be deleted by specifying the entry name or by specifying the IP address which is the only way dynamically added entries can be deleted. This is a sensitive command.

Command Syntax

```
DELETE [ /OUT file-spec/ ] [ ENTRY entry-spec ]
       [ , IPADDRESS  ip-addr ]
```

`OUT` *`file-spec`*

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

`ENTRY` *`entry-spec`*

is the name of the ENTRY object to be deleted. The fully-qualified *`entry-spec`* has the form:

```
$process-name.#entry-name
```

If you specify the SCF object type (ENTRY) or any portion of the object name (*`entry-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`IPADDRESS` *`ip-addr`*

is the IP address of the entry object. The IPADDRESS attribute is used in the cases where an ARP or ATMARP table entry is not named. If a name is supplied, the IPADDRESS attribute is ignored.

Examples

The following examples show how to delete an entry by name and by IP address:

```
1-> DELETE ENTRY $ZTCO.#A1
```

```
1-> ASSUME PROCESS $ZTC0
2-> DELETE ENTRY, IPADDRESS 1.2.3.4
```

Considerations

If an Entry is duplicated in both the ARP and ATMARP tables, the delete command removes both table entries.

DELETE ROUTE Command

The DELETE ROUTE command removes a route from the NonStop TCP/IP subsystem. Only routes in the STOPPED summary state may be deleted.

Command Syntax

```
DELETE [ / OUT file-spec / ] [ ROUTE route-spec]
```

`OUT` *`file-spec`*

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

`ROUTE` *`route-spec`*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *`route-spec`* has the form:

`$`*`process-name`*`.#`*`route-name`*

#RT is reserved; use #R*a...* where *a* is any alpha-numeric character other than T. If you specify the SCF object type (ROUTE) or any portion of the object name (*`route-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command deletes the route #ROU3 from the process $ZTC0:

```
1-> DELETE ROUTE $ZTC0.#ROU3
```

DELETE SERVER Command

The DELETE SERVER command removes the server associated with a SUBNET. This is a sensitive command.

Command Syntax

```
 DELETE [ /OUT file-spec/ ] [ SERVER server-spec ]
```

`OUT` *`file-spec`*

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

`SERVER` *`server-spec`*

is the name of the server. The fully-qualified *`server-spec`* has the form:

`$`*`process-name`*`.#`*`server-name`*

If you specify the SCF object type (SERVER) or any portion of the object name (*`server-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command deletes the server named #SR1:

```
1-> ASSUME PROCESS $ZTC0
2-> DELETE SERVER #SR1
```

Considerations

The SUBNET on which the SERVER object has been assigned must be in the STOPPED state for this command to work.

DELETE SUBNET Command

The DELETE SUBNET command removes a SUBNET from the NonStop TCP/IP subsystem. Only SUBNETs in the STOPPED summary state may be deleted.

Command Syntax

```
DELETE [/ OUT file-spec / ] [ SUBNET subnet-spec ]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`*`.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Examples

The following command deletes all SUBNETs under the process $ZTC0:

```
1-> DELETE SUBNET $ZTC0.*
```

Considerations

- The object-name template (wildcard notation) is supported.
- When the DELETE operation is completed, the definition of the SUBNET you specified for deletion is removed from the subsystem.
- All routes defined through the SUBNET are deleted with the SUBNET.

INFO Command

The INFO command displays the current attribute values for the specified TCP/IP object. Alterable attributes are indicated with an asterisk (*). The INFO command with the detail option is supported only for the PROCESS object.

This is a nonsensitive command.

INFO ADDRMAP Command

The INFO ADDRMAP command displays the current attribute values for a specific address map name entry or for all entries in the NonStop TCP/IP X.25 address table.

Command Syntax

```
INFO [ / OUT file-spec / ] [ ADDRMAP addrmap-spec ]
```

OUT *file-spec*

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

ADDRMAP *addrmap-spec*

defines the NonStop TCP/IP X.25 address table name for which information is to be displayed. The fully-qualified *addrrmap-spec* has the form:

```
$process-name.#addrmap-name
```

If you specify the SCF object type (ADDRMAP) or any portion of the object name (*addrmap-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests information about the specified ADDRMAP:

```
1->INFO ADDRMAP $ZTC0.#A27
```

INFO ADDRMAP Display Format

The format of the display for the INFO ADDRMAP command is:

```
SCF> INFO ADDRMAP $ZTC0.*

TCPIP Info ADDRMAP \SYSA.$ZTC0.*

Name               IPADDRESS                         X121ADDR
#A1                123.456.78.9                      12345678901

                   Last Time Used                    Last Dev/Subdev Used
                   21 Sep 1994, 9:18:23.984          \SYS.$X25A.#TCPIP1

Name               IPADDRESS                         X121ADDR
#A2                123.456.789.0                     1234512345123

                   Last Time Used                    Last Dev/Subdev Used
                   08 Aug 1994, 1:35:16.123          \SYS.$X25B.#TCPIP2
```

`Name`

is the name of the ADDRMAP entry.

`IPADDRESS`

is the IP address.

`X121ADDR`

is the DTE address.

`Last Time Used`

is the date and time this entry was last used.

`Last Dev/Subdev Used`

is the name of the device and subdevice last used for this entry.

INFO ENTRY Command

The INFO ENTRY command displays the ARP or ATMARP table for the given entry. To display the ATM address entries for SVCs and PVCs use the wildcard character *. Entries that have the name ATMentry are ATM address entries.

```
INFO [ /OUT file-spec/ ] [ ENTRY entry-spec ]
                 [ , IPADDRESS  ip addr ]
```

`OUT` *`file-spec`*

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

`ENTRY` *`entry-spec`*

is the name of the entry. The fully-qualified *`entry-spec`* has the form:

`$`*`process-name`*`.#`*`entry-name`*

If you specify the SCF object type (ENTRY) or any portion of the object name (*`entry-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`IPADDRESS` *`ip-addr`*

is the IP address of the entry. The IPADDRESS attribute is used when an ARP entry is not named. If a name is supplied, the IPADDRESS attribute is ignored.

Example

The following commands request information about all entries in the ATMARP, ARP, and ATM address tables:

```
1-> ASSUME PROCESS $ZTC0
2-> INFO ENTRY *
```

INFO ENTRY DISPLAY Format

The format of the display for the INFO ENTRY command table is:

```
TCPIP Info ENTRY \SAMCAT.$ZTCX.*

Name: #A2            (ATMARP)
  IPADDRESS.... 172.16.192.1
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:A1:00

Name: #PVC           (ATMARP PVC)
  IPADDRESS.... 172.16.192.200
  PVC Name..... #IP.PVC001

Name:                (ATMARP)
  IPADDRESS.... 172.16.192.210
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:B2:00

Name:                (ARP)
  IPADDRESS.... 172.16.119.1
  MacAddress... %H00 000C 3920CE

-------------------------------------------------------------
--
Name: ATMentry       (ATM PVC)
  PVC Name..... #ip.pvc001

Name: ATMentry       (ATM SVC)
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:B2:00
```

`Name`

is the name of the entry. The entry type is indicated in parentheses to the right. Possible types are: ATMARP, ATMARP PVC, ATM PVC, ATM SVC, and ARP. ATM address table entries are indicated with the name ATMentry.

`IPADDRESS`

is the IP address for the entry in dotted decimal format.

`PVC Name`

is the name of the PVC associated with the entry.

`AtmAddr`

is the ATM address of the entry.

`MacAddress`

is the MAC (physical) address of the entry in hexadecimal format.

INFO PROCESS Command

The INFO PROCESS command displays the current attribute values for the TCP/IP process object.

Command Syntax

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

   [ , DETAIL ]
```

OUT *file-spec*

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

DETAIL

specifies that the display is to include additional detailed information on the object.

PROCESS *process-name*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Examples

The following command displays the current attributes of the PROCESS object without the DETAIL option:

```
1-> INFO PROCESS $ZTC0
```

The following command displays the current attributes of the PROCESS object with the DETAIL option:

```
1-> INFO PROCESS $ZTCO, DETAIL
```

INFO PROCESS Display Format

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

```
SCF> INFO PROCESS $ZTC0

TCPIP Info PROCESS \SYSA.$ZTC0

*TCPSendSpace  *TCPReceiveSpace  *UDPSendSpace  *UDPReceiveSpace
     8192           8192              8192           8192
```

`TCPSendSpace`

is the amount of data (in bytes) that can be buffered in the TCP layer when sending data to a remote site.

`TCPReceiveSpace`

is the amount of data (in bytes) that can be buffered in the TCP layer when receiving data from a remote site.

`UDPSendSpace`

is the amount of data (in bytes) that can be buffered in the UDP layer when sending data to a remote site.

`UDPReceiveSpace`

is the amount of data (in bytes) that can be buffered in the UDP layer when receiving data from a remote site.

INFO PROCESS Display Format With DETAIL

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

```
SCF> INFO PROCESS $ZTC0, DETAIL

TCPIP Detailed Info PROCESS \SYSA.$ZTC0

*TCP Send Space......... 8192D      *TCP Receive Space...... 8192D
*UDP Send Space......... 8192D      *UDP Receive Space...... 8192D
*Delay Ack Time......... 20         *Delay Ack.............. ON
*Keep Alive Idle........ 45         *Keep Alive Retry Count. 8
*Keep Alive Interval.... 45          QIO Limit..............   100%
*Host ID................ 0.0.0.0
*Host Name.............. sysa.Tandem.COM
 Program Filename....... \SYSA.$SYSTEM.SYS00.TCPIP
*Debug.................. OFF
*Full Dump.............. ON
*All Nets Are Local..... ON
*TCP Compat 42.......... ON
*EXPAND Security........ OFF
*TCP Path MTU........... OFF
*TCP Time Wait.......... 60
Trace Status............ OFF
Trace Filename..........
*ARP Timer Refreshed ... OFF
*RFC1323 Enable ........ ON
*TCP Init Rexmit Timeout 1000  ms
*TCP Min Rexmit Timeout. 400   ms
*TCP Listen Queue Min... 5
*Initial TTL............ 30
```

`TCP Send Space`

is the amount of data (in bytes) that can be buffered in the TCP layer when sending data to a remote site.

`TCP Receive Space`

is the amount of data (in bytes) that can be buffered in the TCP layer when receiving data from a remote site.

`UDP Send Space`

is the amount of data (in bytes) that can be buffered in the UDP layer when sending data to a remote site.

`UDP Receive Space`

is the amount of data (in bytes) that can be buffered in the UDP layer when receiving data from a remote site.

`Delay Ack Time`

is the amount of time (in .01-second units) that acknowledgments are delayed.

`Delay Ack`

indicates whether the acknowledgment (ACK) should be delayed when a TCP packet is received from a remote site.

`Keep Alive Idle`

is the amount of time, in seconds, before TCP issues a keep-alive packet on sockets that have enabled this option.

`Keep Alive Retry Cnt`

is the number of times a keep-alive packet is sent without receiving an acknowledgment. When this value is exceeded, the TCP connection is dropped.

`Keep Alive Interval`

is the time interval, in seconds, between retransmissions of unacknowledged keep-alive packets.

`QIO Limit`

is the value specified for the QIOLIMIT attribute that can be declared as one of the NonStop TCP/IP process startup parameters.

`Host ID`

identifies the host by number.

`Host Name`

is the host on which the TCP/IP process is running.

`Program Filename`

is the name of the file being executed for this process.

`Debug`

is the current setting (ON or OFF) of the DEBUG attribute.

`Full Dump`

is the current setting (ON or OFF) of the FULLDUMP attribute.

`All Nets Are Local`

indicates whether TCP uses the interface MTU as a base for determining the TCP Maximum Segment Size (MSS) for each non-local TCP connection (ON), or whether TCP conforms to RFC-specified behavior and use 512 bytes as the default MSS for non-local segments (OFF).

`TCP Compat 42`

is the flag that sets the NonStop TCP/IP process compatible with BSD4.2 versions, If the flag is ON, then the original ACK minus 1 is sent in the keepalive packet; if the flag is OFF, the original ACK is sent in the keepalive packet.

`EXPAND Security`

indicates whether or not.TCPchecks if a SOCKET request from another Expand node has passed the Expand security check.

`TCP Path MTU`

indicates whether or not TCP uses PATH MTU discovery on all TCP-type sockets (SOCK_STREAM) unless disabled by the SETSOCKOPT for SO_PMTU.

`TCP Time Wait`

is the amount of time in seconds that a TCP connection remains in the TIME_WAIT state. The default is 60 seconds. The range is 1 to 120.

`Trace Status`

is ON when the process is being traced using SCF.

`Trace Filename`

is the name of the current trace file.

`ARP Timer Refreshed`

indicates whether or not TCP restarts the ARP timer every time the ARP table entry is referenced when transmitting an IP packet. The default for this option is OFF. If this option, is enabled stale ARP table entries can take longer to be flushed.

`RFC1323 Enable`

indicates whether or not TCP supports the TCP Large Windows feature, documented in RFC 1323. When you enable this option, the TCP/IP process uses the TCP Window Scale and Timestamp options described in RFC 1323.

`TCP Init Rexmit Timout`

is the initial retransmit timer value to use on a TCP connection.

`TCP Min Rexmit Timeout`

is the minimum value allowed for the TCP retransmission timeout.

`TCP Listem Queue Min`

is the minimum queue length that is set on a TCP socket when the TCP/IP process handles a socket LISTEN or ACCEPT_NW1 function call.

`Initial-TTL`

specifies the initial value for UDP and TCP TTL.

INFO ROUTE Command

The INFO ROUTE command displays attribute values for a specified route routes.

Command Syntax

```
INFO [ / OUT file-spec / ] [ ROUTE route-spec ]
```

OUT *file-spec*

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

ROUTE *route-spec*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *route-spec* has the form:

$*process-name*.#*route-name*

If you specify the SCF object type (ROUTE) or any portion of the object name (*route-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

EXAMPLES

The following command displays the attribute values for a route object named #MR2:

```
1-> INFO ROUTE $ZTCO.#MR2
```

The following command displays the attribute values for all routes subordinate to the PROCESS object $ZTC0:

```
1-> INFO ROUTE $ZTC0.*
```

INFO ROUTE Display Format

The format of the display for the INFO ROUTE command is:

```
SCF> INFO ROUTE *

TCPIP Info ROUTE \SYSA.$ZTC0.*

Name   Subnet  Netmask        Destination     Gateway        Type  Metric

#ROU11 #SN1    255.255.255.0  130.252.153.0    130.252.12.5   G      1
#ROU9  #SN1    255.255.255.0  130.252.154.0    130.252.12.5   G      1
#ROU12 #SN1    255.255.255.0  130.252.131.0    130.252.12.5   G      0
#ROU8  #SN1    255.255.255.0  130.252.11.0     130.252.12.5   G      0
#ROU3 #LOOP0 255.0.0.0    127.0.0.1     127.0.0.1    H     0
```

`Name`

is the name of the route.

`Subnet`

is the name of the SUBNET used by the specific route.

`Netmask`

is the netmask associated with the route entry.

`Destination`

is the IP address of the remote machine or network that can be reached via the IP address shown for Gateway.

`Gateway`

is the IP address of the router or gateway host through which the destination network or host can be reached.

`Type`

indicates one of three types of routes:

`blank`	routes to a network.
`G`	routes to a network through a gateway.
`H`	routes to a host.
`H,G`	routes to a host through a gateway.

`Metric`

indicates the number of hops to the destination.

INFO SERVER Command

The INFO SERVER command displays information about the server. This is a nonsensitive command.

```
INFO [ /OUT file-spec/ ] [ SERVER server-spec ]
```

`OUT` *`file-spec`*

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

`SERVER` *`server-spec`*

is the name of the server. The fully-qualified *`server-spec`* has the form:

```
$process-name.#server-name
```

If you specify the SCF object type (SERVER) or any portion of the object name (*`server-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following commands return information about all currently registered servers.

```
1-> ASSUME PROCESS $ZTC0
2-> INFO SERVER *
```

INFO SERVER Display Format

The display format for INFO SERVER is:

```
TCPIP Info SERVER \SAMCAT.$ZTCY.*

Name: #SRV2       Subnet #EN1
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:A2:00
```

`Name`

is the name of the server.

`AtmAddr`

is the ATM address of the server.

`Subnet`

is the name of the SUBNET to which the server is associated.

INFO SUBNET Command

The INFO SUBNET command displays the current attribute values for the specified SUBNETs.

Command Syntax

```
INFO [ / OUT file-spec / ] [ SUBNET subnet-spec ]
     [, DETAIL ]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`*`.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Examples

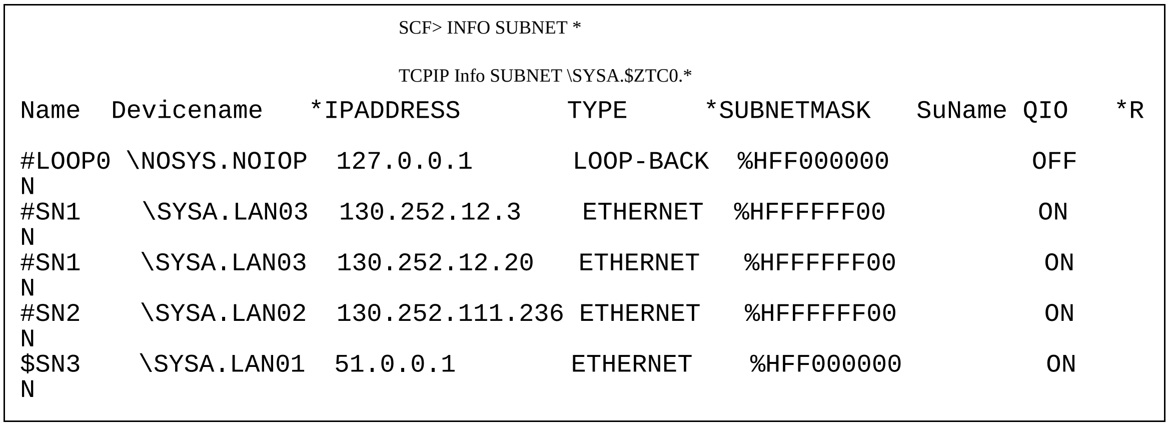
The first example returns information about all running SUBNETs on the system and the second example returns detailed information about a specific SUBNET:

```
1->ASSUME PROCESS $ZTC0
2->INFO SUBNET *

1->INFO SUBNET $ZTC0.#EN3, DETAIL
```

INFO SUBNET Display Format

The format of the display for the INFO SUBNET command is:



```
                              SCF> INFO SUBNET *

                              TCPIP Info SUBNET \SYSA.$ZTC0.*
Name  Devicename   *IPADDRESS       TYPE     *SUBNETMASK   SuName QIO   *R

#LOOP0 \NOSYS.NOIOP  127.0.0.1       LOOP-BACK  %HFF000000          OFF
N
#SN1    \SYSA.LAN03  130.252.12.3    ETHERNET  %HFFFFFF00          ON
N
#SN1    \SYSA.LAN03  130.252.12.20   ETHERNET   %HFFFFFF00          ON
N
#SN2    \SYSA.LAN02  130.252.111.236 ETHERNET   %HFFFFFF00          ON
N
$SN3    \SYSA.LAN01  51.0.0.1        ETHERNET    %HFF000000          ON
N
```

`Name`

is the name of the SUBNET. Note that in this example, #SN1 has an alias, listed right below it. The IPADDRESS of the varies by the final digit only.

`Devicename`

is the name of the SLSA LIF that provides access to the Ethernet LAN, is the name of the ATM adapter, or is the name of the X25AM line that provides access to the X.25 network. Note that with loopback SUBNETs, the value \NOSYS.NOIOP, meaning no system, no device name, is displayed. This value is displayed because loopback SUBNETs are routed internally so that there is no device name to display.

`IPADDRESS`

is the Internet address of this SUBNET and all the IP addresses of the aliases associated with the SUBNET.

`TYPE`

is the SUBNET type. It can be LOOPBACK, ATM, ETHERNET, SNAP or X.25.

`SUBNETMASK`

is a 32-bit integer that specifies which portion of the network number and the IP host address is to be masked to define a subnet.

`SuName`

is the name of the X.25 subdevice used by the SUBNET.

`QIO`

shows whether or not the SUBNET is currently using the QIO interface. QIO is always on for Ethernet, ATM and SNAP type SUBNETs. (This attribute is not applicable for X25AM SUBNETs.) ON indicates that the interface is currently using QIO mode. OFF indicates that the interface is not currently using QIO mode.

`R`

shows whether or not the ICMP Router Discovery Protocol (IRDP) has been enabled on the SUBNET. The displayed value can be Y (IRDP is ON), or N (IRDP is OFF).

INFO SUBNET Display Format With Detail for ATM

The format of the INFO SUBNET with the detail option selected for an ATM SUBNET is:

```
TCPIP Info SUBNET \NEWYORK.$ZTC0.#EN3, DETAIL

Name  Devicename    *IPAddRESS  TYPE  *SUBNETMASK SuName QIO
*R
#EN3  \NEWYORK.$LAN03  51.0.0.1  ATM   %HFFFFFF00        ON
N
Trace Status ........ OFF
Trace Filename ......
Interface MTU ....... 9180
Gateway.............. OFF
AtmAddr...
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:B2:01
ATMARP Server Mode .. Server
ATM SELECTOR BYTE ... 1
```

`Name`

is the name of the SUBNET. Note that in this example, #SN1 has an alias, listed right below it. The IPADDRESS of the varies by the final digit only.

`Devicename`

is the name of the SLSA LIF that provides access to the Ethernet LAN, is the name of the ATM adapter, or is the name of the X25AM line that provides access to the X.25 network. Note that with loopback SUBNETs, the value \NOSYS.NOIOP, meaning no system, no device name, is displayed. This value is displayed because loopback SUBNETs are routed internally so that there is no device name to display.

`IPADDRESS`

is the Internet address of this SUBNET and all the IP addresses of the aliases associated with the SUBNET.

`TYPE`

is the SUBNET type. It can be LOOPBACK, ATM, ETHERNET, SNAP or X.25.

`SUBNETMASK`

is a 32-bit integer that specifies which portion of the network number and the IP host address is to be masked to define a SUBNET.

`SuName`

is the name of the X.25 subdevice used by the SUBNET.

`QIO`

shows whether or not the SUBNET is currently using the QIO interface. QIO is always on for Ethernet, ATM and SNAP type SUBNETs. (This attribute is not applicable for X25AM SUBNETs.) ON indicates that the interface is currently using QIO mode. OFF indicates that the interface is not currently using QIO mode.

`R`

shows whether or not the ICMP Router Discovery Protocol (IRDP) has been enabled on the SUBNET. The displayed value can be Y (IRDP is ON), or N (IRDP is OFF).

`Trace Status`

is ON if the SUBNET is being traced with SCF.

`Trace Filename`

is the name of the current trace file.

`Interface MTU`

is the maximum transmission unit that can be used on the SUBNET.

`GATEWAY`

directs the SUBNET to add a default filter on the LIF when set to ON. When an IP packet comes in that does not match any filter on the LIF, the LIF routes the packet to the SUBNET owning the default filter (that is, with GATEWAY ON). Only one SUBNET on a LIF can own the default filter. The default is OFF.

`AtmAddr`

is the ATM address assigned to the ATM adapter being used by the SUBNET.

`ATMARP Server Mode`

indicates whether the SUBNET has been configured to act as an ATMARP server or client.

`ATM SELECTOR BYTE`

is the value of the selector byte the SUBNET uses.

INFO SUBNET Display With Detail for Ethernet

The format of the INFO SUBNET with the detail option selected for an Ethernet SUBNET is:

```
TCPIP Detailed Info SUBNET \NEWYORK.$ZTC4.*

Name  Devicename *IPAddRESS  TYPE  *SUBNETMASK SuName QIO *R

#EN1\NY.$LAN03 172.17.214.234 ETHERNET %HFFFFFF00     OFF  N
Trace Status ........ OFF
Trace Filename ......
Interface MTU ....... 1500
Gateway.............. OFF
---Multicast Groups--- ---State---
   224.0.0.1             STARTED
   230.17.123.55         STARTING
   239.1.2.3             STOPPED
```

`Name`

is the name of the SUBNET. Note that in this example, #SN1 has an alias, listed right below it. The IPADDRESS of the varies by the final digit only.

`Devicename`

is the name of the SLSA LIF that provides access to the Ethernet LAN, is the name of the ATM adapter or is the name of the X25AM line that provides access to the X.25 network. Note that with loopback SUBNETs, the value \NOSYS.NOIOP, meaning no system, no device name, is displayed. This value is displayed because loopback SUBNETs are routed internally so that there is no device name to display.

`IPADDRESS`

is the Internet address of this SUBNET and all the IP addresses of the aliases associated with the SUBNET.

`TYPE`

is the SUBNET type. It can be LOOPBACK, ATM, ETHERNET, SNAP or X.25.

`SUBNETMASK`

is a 32-bit integer that specifies which portion of the network number and the IP host address is to be masked to define a subnet.

`SuName`

is the name of the X.25 subdevice used by the SUBNET.

`QIO`

shows whether or not the SUBNET is currently using the QIO interface. QIO is always on for Ethernet, ATM and SNAP type SUBNETs. (This attribute is not applicable for X25AM SUBNETs.) ON indicates that the interface is currently using QIO mode. OFF indicates that the interface is not currently using QIO mode.

`R`

shows whether or not the ICMP Router Discovery Protocol (IRDP) has been enabled on the SUBNET. The displayed value can be Y (IRDP is ON), or N (IRDP is OFF).

`Trace Status`

is ON if the SUBNET is being traced with SCF.

`Trace Filename`

is the name of the current trace file.

`Interface MTU`

is the maximum transmission unit that can be used on the SUBNET.

`GATEWAY`

makes the TCP/IP process the gateway for routing IP traffic between multiple subnets when set to ON.

`Multicast Groups`

indicates the IP multicast group addresses that the TCP/IP process is listening to.

`State`

is the current state of the socket; it applies only to sockets whose Proto value is TCP. The possible values are:

`STARTED`

indicates that multicast is operational for the group.

STARTING

indicates that the multicast group is transitioning to the STARTED (and operational) state but is not yet fully operational.

STOPPED

indicates that multicast is not operational for the group.

Considerations

- The object-name template (wildcard notation) is supported.
- The STATUS and STATS commands provide information on the summary state and the statistics of TCP/IP objects.

LISTOPENS Command

The LISTOPENS command returns information on openers of the TCP/IP process.

This is a nonsensitive command.

LISTOPENS PROCESS Command

Command Syntax

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

   [ , DETAIL ]
```

OUT *file-spec*

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

PROCESS *process-name*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

DETAIL

specifies that the display is to include additional detailed information on the object.

Examples

The following command requests non-detailed and detailed information about the openers of the specified process:

```
1->LISTOPENS PROCESS $ZTC0
1->LISTOPENS PROCESS $ZTC0,DETAIL
```

LISTOPENS PROCESS Display Format

The format of the display for the LISTOPENS command without the DETAIL option is:

```
SCF> LISTOPENS PROCESS $ZTC0

TCPIP Listopens PROCESS \SYSA.$ZTC0.*

Openers             PPID      BPID     PLFN   BLFN   Protocol   Lport
$ZNET               0,25               3      0      #ZSPI      *
$ZTNT               13,39              5      0      TCP        telnet
$Z068               12,70              3      0      TCP        4241
$ZPORT              4,66               6      0      TCP        ftp
$ZPORT              4,66               5      0      TCP        finger
$ZPORT              4,66               4      0      TCP        echo
$ZTNT               13,29              5      0      TCP        telnet
$ZTNT               3,83               4      0      TCP        telnet
$ZTNT               3,83               3      0      TCP        telnet
$ZTNT               13,29              3      0      TCP        telnet
```

`Openers`

is the process name of the opener of the TCP/IP process.

`PPID`

is the primary processor and PIN of the opener.

`BPID`

is the backup processor and PIN of the opener.

`PLFN`

is the logical file number of the primary opener process.

`BLFN`

is the logical file number of the backup opener process.

`Protocol`

is the protocol accessed by the opener.

`Lport`

is the local port number for either TCP or UDP, depending on the value of Protocol. The more common port values are displayed in text form; others are displayed as four-decimal octets.

LISTOPENS Display Format With DETAIL

The format of the display for the LISTOPENS command with the DETAIL option is a combination of the displays for the LISTOPENS and STATUS PROCESS commands:

```
SCF> LISTOPENS PROCESS $ZTC0,DETAIL

TCPIP Detailed Listopens PROCESS \SYSA.$ZTC0.*

Opener $ZTNT            Ppid    0,279    Bpid           Plfn  3      Blfn 0
       Proto  TCP       State   LISTEN        SendQ      0    RecvQ      0
       Laddr  0.0.0.0           Lport  telnet
       Faddr  0.0.0.0           Fport  *

Opener $ZPORT           Ppid    0,280    Bpid           Plfn  3      Blfn 0
       Proto  TCP       State   LISTEN        SendQ      0    RecvQ      0
       Laddr  0.0.0.0           Lport  echo
       Faddr  0.0.0.0           Fport  *

Opener $ZPORT           Ppid    0,280    Bpid           Plfn  3      Blfn 0
       Proto  TCP       State   LISTEN        SendQ      0    RecvQ      0
       Laddr  0.0.0.0           Lport  finger
       Faddr  0.0.0.0           Fport  *

Opener $ZPORT           Ppid    0,280    Bpid           Plfn  3      Blfn 0
       Proto  TCP       State   LISTEN        SendQ      0    RecvQ      0
       Laddr  0.0.0.0           Lport  ftp
       Faddr  0.0.0.0           Fport  *
```

`Opener`

is the process name of the opener of the TCP/IP process.

`Ppid`

is the primary processor and PIN of the opener.

`Bpid`

is the backup processor and PIN of the opener.

`Plfn`

is the logical file number of the primary opener process.

`Blfn`

is the logical file number of the backup opener process.

`Proto`

is the protocol associated with the socket, which can be UDP (for a UDP socket), TCP (for a TCP socket), or a protocol number (for a raw IP socket).

`State`

is the current state of the socket; it applies only to sockets whose Proto value is TCP. The possible values are:

`CLOSING`

if waiting for a terminate connection request acknowledgment from the remote site.

`CLOSE-WAIT`

if waiting for a terminate connection request from the local user.

`ESTAB`

if the connection is open and the user can send and receive data. This is the normal state for data transfer.

`FIN-WAIT-1`

if waiting for a terminate connection request from the remote TCP site or if waiting for acknowledgment of the terminate connection request that the process has sent previously.

`FIN-WAIT-2`

if waiting for a termination of data to be received after having sent a FIN (termination of data being sent).

`LAST-ACK`

if waiting for acknowledgment of the terminate connection request previously sent to the remote site (which includes an acknowledgment of its terminate connection request).

`LISTEN`

if waiting for a connection request from any remote TCP site.

`SYN-RCVD`

if waiting for an acknowledgment of a SYN-ACK sent in response to a SYN.

`SYN-SENT`

if waiting for a SYN-ACK after having sent a SYN.

`TIME-WAIT`

if waiting for sufficient time to pass (about two round trips) to be sure that stray packets are flushed from the network.

`Laddr`

is the local Internet address associated with the socket, displayed as four-decimal octets.

`Lport`

is the local port number for either TCP or UDP, depending on the value of Proto. The more common port values are displayed in text form; others are displayed as four-decimal octets.

`Faddr`

is the foreign (remote) Internet address associated with the socket, displayed in four-decimal octets.

`Fport`

is the foreign port number for either TCP or UDP, depending on the value of Proto. The more common port values are displayed in text form; others are displayed as four-decimal octets.

`SendQ`

is the number of bytes of data in the send queue of the socket.

`RecvQ`

is the number of bytes of data in the receive queue of the socket.

Considerations

The object-name template (wildcard notation) is supported.

NAMES Command

The NAMES command displays the names of the specified TCP/IP objects. The object types can be ADDRMAP, ROUTE, or SUBNET.

This is a nonsensitive command.

NAMES ADDRMAP Command

The NAMES ADDRMAP command displays the names of the address maps for a specific TCP/IP process.

Command Syntax

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

`OUT` *`file-spec`*

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

`ADDRMAP` *`process-name`*

specifies the name of the TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests the names of the ADDRMAP objects on the system:

```
1->NAMES ADDRMAP $ZTC0.*
```

NAMES ADDRMAP Display Format

The format of the display for the NAMES ADDRMAP command is:

```
TCPIP Names ADDRMAP \SYSA.$ZTC0.*

ADDRMAP
#A1                #A2              #A3
```

NAMES ROUTE Command

The NAMES ROUTE command displays the names of the routes for the NonStop TCP/IP subsystem.

Command Syntax

```
NAMES [ / OUT file-spec / ] [ ROUTE route-spec ]
```

`OUT` *`file-spec`*

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

`ROUTE` *`route-spec`*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *`route-spec`* has the form:

`$`*`process-name`*`.#`*`route-name`*

If you specify the SCF object type (ROUTE) or any portion of the object name (*`route-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests a list of the routes associated with $ZTC0:

```
1-> NAMES ROUTE $ZTC0.*
```

NAMES ROUTE Display Format

The format of the display for the NAMES ROUTE command is:

```
TCPIP Names ROUTE \SYSA.$ZTC0.*

ROUTE
#ROU11    #ROU9    #ROU12    #ROU8    #ROU3
```

NAMES SUBNET Command

The NAMES SUBNET command displays the names of specified SUBNETs.

Command Syntax

```
NAMES [ / OUT file-spec / ] [ SUBNET subnet-spec ]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`* `.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests the names of the SUBNETs associated with $ZTC0:

```
SCF> NAMES SUBNET $ZTC0.*
```

NAMES SUBNET Display Format

The format of the display for the NAMES SUBNET command is:

```
TCPIP Names SUBNET \SYSA.$ZTC0.*

SUBNET
#LOOP0  #EN1  #EN2
```

Considerations

The object-name template (wildcard notation) is supported.

PRIMARY Command

The PRIMARY command can be used when the NonStop TCP/IP subsystem is running as a NonStop process pair. This command causes the backup processor to become the primary processor and the primary processor to become the backup processor. This is a sensitive command.

PRIMARY PROCESS Command

Command Syntax

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

   , CPU cpu-number
```

`OUT` *`file-spec`*

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

`PROCESS` *`process-name`*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`CPU` *`cpu-number`*

is the number of the processor of the backup process. This attribute is required.

Examples

The following command causes the $ZTC0 primary process to become the backup process (the attribute processor 6 identifies the processor where the former backup process resided):

```
1-> PRIMARY PROCESS $ZTC0, CPU 6
```

Considerations

Run the NonStop TCP/IP process as a NonStop process pair. The NonStop TCP/IP process can only access the LAN hardware when running in the processors that access the ServerNet addressable controller (SAC)

START Command

The START command initiates the operation of SUBNETs and routes for the NonStop TCP/IP subsystem. When the subsystem has successfully completed processing this command, the specified object is placed in the STARTED summary state. You can start a SUBNET or a route, but not a process. If a process is not started, it is undefined.

This is a sensitive command.

START ROUTE Command

The START ROUTE command creates implicit connections to and from a route. The successful completion of the START command leaves the route in the STARTED summary state.

Command Syntax

```
START [ / OUT file-spec / ] [ ROUTE route-spec ]
```

OUT *file-spec*

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

ROUTE *route-spec*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *route-spec* has the form:

$*process-name*.#*route-name*

If you specify the SCF object type (ROUTE) or any portion of the object name (*route-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following commands start all routes under the assumed process:

```
1-> ASSUME PROCESS $ZTC0
2-> START ROUTE *
```

START SUBNET Command

The START SUBNET command creates implicit connections to and from a SUBNET. The SUBNET transitions through the STARTING state and, upon successful completion, ends in the STARTED summary state.

Command Syntax

```
START [ / OUT file-spec / ] [ SUBNET subnet-spec ]
```

OUT *file-spec*

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

SUBNET *subnet-spec*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *subnet-spec* has the form:

$*process-name*.#*subnet-name*

If you specify the SCF object type (SUBNET) or any portion of the object name (*subnet-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following commands start SUBNET #SN1 under the assumed process:

```
1-> ASSUME PROCESS $ZTC0
2-> START SUBNET #SN1
```

Considerations

- The object-name template (wildcard notation) is supported.
- When you use the START command, the object must be in the STOPPED summary state.
- The SLSA subsystem must be operational before SUBNETs can be started successfully.
- To terminate the operation of SUBNETs, use the STOP or ABORT command.

STATS Command

The STATS command returns statistics for a specified TCP/IP object. All statistics are 32-bit numbers. The letter D in the display values indicates that the value is a doubleword.

Whenever a RESET option is included, the counters associated with the specified objects are displayed and reset to 0, and the timestamp for the reset is recorded. Any STATS command returns the time at which the current statistics were sampled and the time at which the counters were last reset.

This is a sensitive command when used with the reset option.

STATS ADDRMAP Command

The STATS ADDRMAP command returns statistics for a specific address name entry in the NonStop TCP/IP X.25 address table.

Command Syntax

```
STATS [ / OUT file-spec / ] [ ADDRMAP addrmap-spec ]

   [ , RESET ]
```

`OUT` *`file-spec`*

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

`ADDRMAP` *`addrmap-spec`*

defines the entry name in the NonStop TCP/IP X.25 address table for which statistics are requested. The fully-qualified *`addrmap-spec`* has the form:

$*process-name* .#*addrmap-name*

If you specify the SCF object type (ADDRMAP) or any portion of the object name (*`addrmap-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests statistics for the specified ADDRMAP:

```
1->STATS ADDRMAP $ZTC0.*
```

STATS ADDRMAP Display Format

The format of the display for the STATS ADDRMAP command is:

```
SCF> STATS ADDRMAP $ZTC0.*

TCPIP Stats ADDRMAP \SYS.$ZTC0.*

Sample Time..  08 SEP 1996, 10:15:04.541
Reset Time...  08 SEP 1996, 09:10:03.123

Name        IPADDRESS         X121ADDR            Call-Out    Call-In

#DTE        123.456.78.9      12345678901234         3           1

                   Last Time Used
                   08 SEP 1996, 09:12:16.123
```

`Sample Time`

is the date and time when the statistics were sampled.

`Reset Time`

is the date and time when the counters were last set to 0.

`Name`

is the ADDRMAP entry name in the NonStop TCP/IP X.25 address table.

`IPADDRESS`

is the IP address.

`X121ADDR`

is the DTE address.

`Call-Out`

is the number of times calls were placed.

`Call-In`

is the number of times calls were received.

`Last Time Used`

is the date and time the DTE address was last used.

STATS PROCESS Command

The STATS PROCESS command displays the NonStop TCP/IP subsystem statistics for each of the protocol layers.

Command Syntax

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

   [ , RESET ]
```

`OUT` *`file-spec`*

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

`PROCESS` *`process-name`*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`RESET`

resets the statistical counters to zero.

Examples

The following commands requests statistics about the specified process:

```
1-> STATS PROCESS $ZTC0
```

```
1-> ASSUME PROCESS $ZTC0
2-> STATS
```

STATS PROCESS Display Format

The format of the display for the STATS PROCESS command is:

```
TCPIP Stats PROCESS \SYSA.$ZTC0

Sample Time ... 17 Oct 1996, 17:17:41.169
Reset Time .... 28 Sep 1996, 17:08:04.488

                               TCP LAYER STATS
Bad Checksum.......... 0D               Bad Offset............ 3D
Invalid Header Size... 0D               Bad Segment Size...... 0D
Retransmitted Packets. 608D             Connection Timeouts... 129D
Total Packets Input... 103579D          Total Packets Output.. 9847D
Incoming Connections.. 804D             Outgoing Connections.. 1096D
No Ports For Packets.. 9D               Urgent Packets Recv... 3D
Packets Unacknowledged 0D               Established Connects.. 1D
Connections Dropped... 0D               Embryonic Conn Dropped 0D
Connections Closed.... 0D               Segments RTT.......... 0D
RTT Updated........... 0D               Delayed ACKs Sent..... 0D
Conn Dropped Timeouts. 0D               Retransmit Timeouts... 0D
Persist Timeouts...... 0D               Keep-Alive Timeouts... 0D
Keep-Alive Probes Sent 0D               Keep-Alive Dropped.... 0D
Data Packets Sent..... 788D             Data Bytes Sent....... 67897D
Retransmitted Bytes... 17890D           ACK Packets Sent...... 1298D
Window Probes Sent.... 0D               Urgent Packets Sent... 0D
Window Update PKT Sent 0D               Control Packets Sent.. 0D
Data Packets Received. 344D             Data Bytes Received... 34489D
Duplicate PKTs Recv... 0D               Duplicate Bytes Recv.. 0D
Partial Duplicate PKTs 0D               Partial Duplicate Byte 0D
Out Of Order PKTs Recv 0D               Out Of Order Byte Recv 0D
PKTs Recv After Window 0D               Bytes Recv After Win.. 0D
PKTs Recv After Close. 0D               Window Probe PKTs Recv 0D
Duplicate ACKs Recv... 0D               Too Much ACK Received. 0D
ACK Packets Received.. 0D               ACK Bytes Received.... 0D
Window Update PKTs.... 0D               ACK Predictions OK.... 110D
Data Predictions OK... 501D             SYN Cache Added....... 0D
SYN Cache Completed... 0D               SYN Cache Timed Out... 0D
SYN-C Dropped, Overflow0D               SYN-C Dropped, RST.... 0D
SYN-C Dropped, Unreach.0D               SYN-C Dropped, Buktflow0D
SYN Cache Aborted .... 0D               SYN Cache Duplicated.. 0D
SYN Cache Dropped..... 0D               SYN Drops,Queue full.. 0D
Bad SYN Packets....... 1D

                              UDP LAYER STATS
Bad Checksum.......... 0D
Invalid Header Size... 0D               Bad Packet Size....... 0D
Total Packets Input... 0D               Total Packets Output.. 0D
Input Packets Dropped  OD               Output Packets Dropped OD

                               IP LAYER STATS
Bad Checksum.......... 0D
Invalid Header Size... 0D               Bad Packet Size....... 0D
Fragments Input....... 0D               Fragments Dropped..... 0D
Packets Cant Forward.. 3D               ICMP Redirect Sent.... 0D
Short Packets......... 0D               Packets Too Small..... 0D
Fragments Timed Out... 0D               Packets Forwarded..... 0D
Total Packets Input... 12327D           Total Packets Output.. 8129D

                              IP ROUTING STATS
Bad Route Redirects... 0D               Dynamic Redirects..... 0D
New Gateway Redirects. 0D               WildCard Matches...... 0D
Unreachable........... 10D
```

The STATS PROCESS command display (continued):

```
                              ICMP LAYER STATS
Bad Checksum.......... 0D               Errors................ 4248D
Invalid Header Size... 0D               Short IP Packets...... 0D
Reflect Packets....... 4D               Bad ICMP Packets...... 0D
Bad ICMP Code......... 1D               Packets Too Short..... 0D
In Echo Reply......... 0D               Out Echo Reply........ 4D
In Dest Unreachable... 643D             Out Dest Unreachable.. 4248D
In Source Quench...... 3D               Out Source Quench..... 0D
In Redirect........... 0D               Out Redirect.......... 0D
In Echo............... 0D               Out Echo.............. 0D
In Time Exceeded...... 1D               Out Time Exceeded..... 0D
In Parameter Problem.. 0D               Out Parameter Problem. 0D
In Timestamp.......... 0D               Out Timestamp......... 0D
In Timestamp Reply.... 0D               Out Timestamp Reply... 0D
In Info Request....... 0D               Out Info Request...... 0D
In Info Reply......... 0D               Out Info Reply........ 0D
Bad Router Adv Subcode 0D               Bad Router Addr List.. 0D
Bad Router Words/Addr. 0D               Good Routes Recorded.. 0D
Router Advertisement.. 0D               Router Solicitation... 0D

                               QIO STATS
Data MDs In Use........ 0D              Maximum Data MDs Used.  1D
Dup MDs In Use......... 0D              Maximum Dup MDs Used..  0D
Dup Driver MDs In Use.. 0D              Max Dup Driv MDs Used.  0D
No Data MDs Avail...... 0D              No Dup MDs Avail......  0D
MD Queue Limits........ 0D              QIO Limit Warnings....  0D
QIO Driver Errors...... 0D              No Dup Driv MDs Avail.  0D
Current Pool Allocation 452156D         Maximum Pool Allocation 452156D
Pool Allocation Fails.. 0D
Total MBUFs Allocated.. 1008D           Current MBUFs Used....  8D
Maximum MBUFs Used..... 14D             MBUF Allocation Fails.  0D

                              SOCKET SEND SIZE HISTOGRAM
Size 1-128............. 7D              Size 129-256........... 0D
Size 257-512........... 0D              Size 513-1024.......... 0D
Size 1025-2048......... 0D              Size 2049-4096......... 0D
Size 4097-8192......... 0D              Size 8193-12288........ 0D
Size 12289-16384....... 0D              Size 16385-32768....... 0D

                              ARP STATS
In ARP Requests........ 0D              Out ARP Requests....... 5D
In ARP Replys.......... 4D              Out ARP Replys......... 0D
In InARP Requests...... 2D              Out InARP Requests..... 2D
In InARP Replys........ 1D              Out InARP Replys....... 2D
In ARP Naks............ 1D              Out ARP Naks........... 0D

                              IGMP STATS
Total Packets Input.... 0D               Total Reports Sent.... 0D
Short Packets.......... 0D               Bad Checksum.......... 0D
Total Queries Input.... 0D               Bad Queries........... 0D
Total Reports Input.... 0D               Bad Reports........... 0D
Reports For Our Groups. 0D
```

Statistics Definitions (in Alphabetical Order)

`Reset Time`

is the time at which the counters were last initialized (set to zero).

`Sample Time`

is the time at which the statistics were sampled.

Description of Statistics for the TCP Layer

`ACK Bytes Received`

is the number of ACK bytes acknowledged by received ACKs.

`ACK Packets Received`

is the number of ACK packets received.

`ACK Packets Sent`

is the number of ACK packets sent.

`ACK Predictions OK`

is the number of times the header predictions were correct for ACKs.

`Bad Checksum`

is the number of packets received with invalid checksum values.

`Bad Offset`

is the number of packets received with invalid data offsets in their TCP headers. An invalid data offset usually indicates that either the sender of the packet made an internal error in generating the packet, or the receiver of the packet had a byte-swapping problem. This error is rare and is usually seen only during the development of the protocol.

`Bad Segment Size`

is the number of packets received with invalid segment sizes.

`Bad SYN Packets`

is the number of bad SYN packets.

`Bytes Recv After Win`

is the number of bytes received exceeding the window boundary.

`Conn Dropped Timeouts`

is the number of connections dropped in a transmit timeout.

`Connection Timeouts`

is the number of connections (including partial connections) that timed out.
A connection timeout is recorded each time the keep-alive timer or retransmission timer expires. The keep-alive timer expires when the connection is inactive for a certain period of time. The inactivity can be caused by a lost connection or by

network congestion. The retransmission timer expires when a packet is not acknowledged within a certain time.

Packet retransmission can be caused by any of the following conditions: the network is overloaded; the other end of the connection is overloaded (so that appropriate acknowledgments cannot be received and/or sent); or a corrupted packet (that is, a packet with an invalid checksum) is received.

`Connections Closed`

is the number of connections closed (this value includes the number of connections dropped).

`Connections Dropped`

is the number of connections dropped.

`Control Packets Sent`

is the number of SYN, FIN, and RST control packets sent.

`Data Bytes Received`

is the number of bytes received in sequence.

`Data Bytes Sent`

is the total number of data bytes sent.

`Data Packets Received`

is the number of packets received in sequence.

`Data Packets Sent`

is the total number of data packets sent.

`Data Predictions OK`

is the number of times the header predictions were correct for data packets.

`Delayed ACKs Sent`

is the number of delayed ACKs sent.

`Duplicate ACKs Recv`

is the number of duplicate ACK packets received.

`Duplicate Bytes Recv`

is the number of duplicate bytes received.

`Duplicate PKTs Recv`

is the number of duplicate packets received.

`Embryonic Conn Dropped`

is the number of embryonic connections dropped.

`Established Connects`

is the number of connections established.

`Incoming Connections`

is the number of incoming connection requests.

`Invalid Header Size`

is the number of packets received with an invalid header size. This error usually indicates a problem between IP and TCP.

`Keep-Alive Dropped`

is the number of connections dropped because of keep-alive timeouts.

`Keep-Alive Probes Sent`

is the number of keep-alive probes sent.

`Keep-Alive Timeouts`

is the number of keep-alive timeouts.

`No Ports For Packets`

is the number of packets received for a connection that has been closed or does not exist. This event can be a normal occurrence or it can be caused by a faulty TCP/IP implementation that does not conform to the TCP/IP state table.

`Outgoing Connections`

is the number of connection requests sent to remote hosts.

`Out Of Order PKTs Recv`

is the number of out-of-order packets received.

`Out Of Order Byte Recv`

is the number of out-of-order bytes received.

`Packets Unacknowledged`

is the number of unacknowledged packets.

`Partial Duplicate Byte`

is the number of duplicate bytes received in partially duplicate packets.

`Partial Duplicate PKTs`

is the number of packets received with some duplicate data.

`Persist Timeouts`

is the number of persistent timeouts.

`PKTs Recv After Close`

is the number of packets received after close.

`PKTs Recv After Window`

is the number of packets received exceeding the window boundary.

`Retransmitted Bytes`

is the number of bytes retransmitted.

`Retransmitted Packets`

is the number of packets retransmitted. Packets are retransmitted when a packet is not acknowledged within a certain time period. Packets can be retransmitted for any of the following reasons: the network is overloaded; the other end of the connection is overloaded (so that appropriate acknowledgments cannot be received or sent); or a corrupted packet (that is, a packet with an invalid checksum) has been received.

`Retransmit Timeouts`

is the number of retransmit timeouts.

`RTT Updated`

is the number of round-trip times updated.

`SYN Cache Added`

is the number of SYN cache entries added.

`SYN Cache Completed`

is the number of SYN cache connections completed.

`SYN Cache Timed Out`

is the number of SYN cache entries timed out.

`SYN-C dropped, Overflow`

is the number of SYN cache entries dropped because of overflow.

`SYN-C dropped, RST`

is the number of SYN cache entries dropped because of RST.

`SYN-C dropped, Unreach`

is the number of SYN cache entries dropped because ICMP is unreachable.

`SYN-C dropped, Buktflow`

is the number of SYN cache entries dropped because of bucket overflow.

`SYN Cache Aborted`

is the number of SYN cache aborted (no memory).

`SYN Cache Duplicated`

is the number of duplicated SYNs received.

`SYN Cache Dropped`

is the number of SYNs dropped (no route or memory).

`SYN Drops, Queue full`

is the number of SYN request dropped when SYN queue is full.

`Segments RTT`

is the number of segments where round-trip time attempted.

`Too Much ACK Received`

is the number of ACK packets received for unsent data.

`Total Packets Input`

is the number of packets received.

`Total Packets Output`

is the number of packets sent down to the IP layer.

`Urgent Packets Recv`

is the number of packets received with the URG bit set.

`Urgent Packets Sent`

is the number of packets sent with the URG bit set.

`Window Probes Sent`

is the number of window probes sent.

`Window Update PKT Sent`

is the number of window update packets sent.

`Window Probe PKTs Recv`

is the number of window-probes packets received.

`Window Update Pkts`

is the number of window update packets received.

Description of Statistics for the UDP Layer

`Bad Checksum`

is the number of packets received with invalid checksum values. An invalid checksum is usually caused by a noisy link.

`Bad Packet Size`

is the number of packets received that contain either more or less data than has been specified in their headers. This error indicates the sender has a protocol error or that the receiver has a byte-ordering problem.

`Input Packets Dropped`

is the number of packets not forwarded to socket applications because of receive socket space being full.

`Invalid Header Size`

is the number of packets received with invalid header size. This error indicates a problem between IP and UDP.

`Output Packets Dropped`

is the number of packets not sent because of interface problems.

`Total Packets Input`

is the number of packets received.

`Total Packets Output`

is the number of packets sent to the IP layer.

Description of Statistics for the IP Layer

`Bad Checksum`

is the number of packets received with invalid checksum values. An invalid checksum is usually caused by a noisy link.

`Bad Packet Size`

is the number of packets received with a packet length shorter than expected. This error is very similar to the Invalid Header Size and is usually caused by similar conditions.

`Fragments Dropped`

is the number of packet fragments dropped. A fragment is dropped either when memory cannot be allocated for the fragment or when the fragment is a duplicate of a fragment that has already been received.

`Fragments Input`

is the number of packet fragments received. Usually, a packet is fragmented when it is too large for a particular gateway or network. This statistic may indicate that the sender's maximum segment size is too large for the connection.

`Fragments Timed Out`

is the number of packet fragments received that timed out before the whole packet was received. This is usually caused by congestion, noisy links, or some event that prevents one of the fragments from being received with the rest.

`ICMP Redirects Sent`

is the number of ICMP Redirect messages sent. Redirect messages are sent to the source host to indicate that there is a shorter path to the destination. The source host should send the packet directly to the destination host or to another gateway or router.

`Invalid Header Size`

is the number of packets received with a header size that is larger than the header length provided in the packet. This error indicates a problem with the sender of the packet or a problem in reading the data from the link controller to IP.

`Packets Cant Forward`

is the number of packets destined for another host that were received but could not be forwarded. The packets could not be forwarded because either the local host is not configured as a gateway or no route is available to the specified destination.

`Packets Forwarded`

is the number of packets destined for another host that were forwarded.

`Packets Too Small`

is the number of packets that contained less data than was expected when the packet was read into the local buffers. This error usually indicates a problem with the local machine's buffering scheme.

`Short Packets`

is the number of packets that contained less data than specified in their header. This can be caused by noisy links, a protocol error by the sender of the packet, or a byte-swapping problem on the receiver.

`Total Packets Input`

is the number of packets received.

`Total Packets Output`

is the number of packets sent to the IP layer.

Description of Statistics for IP Routing

`Bad Route Redirects`

is the number of Redirect messages received.

`Dynamic Redirect`

is the number of dynamic route messages received. These messages indicate where the NonStop TCP/IP subsystem should route messages for a specific destination.

`New Gateway Redirects`

is the number of messages received that established a route for a new or an unknown gateway.

`Unreachable`

is the number of messages received that indicated that the specified destination was unreachable.

`WildCard Matches`

is the number of wildcard matches found when zeros were given in the destination Internet address for a route.

Description of Statistics for the ICMP Layer

`Bad Checksum`

is the number of packets received with invalid checksum values. An invalid checksum is usually caused by a noisy link.

`Bad ICMP Code`

is the number of packets received that contain invalid ICMP packet-type codes in the header. The NonStop TCP/IP subsystem supports the following ICMP packet types and packet-type code:

```
Echo Reply (0)
Destination Unreachable (3)
Source Quench (4)
Redirect (5)
Echo (8)
Time Exceeded (11)
Parameter Problem (12)
Timestamp (13)
Timestamp Reply (14)
Information Request (15)
Information Reply (1
```

For more detailed descriptions of these packet types, refer to the descriptions of the individual packet types below.

`Bad ICMP Packets`

is the number of invalid ICMP packets received.

`Bad Router ADDR List`

is the number of IRDP messages with a bad address list.

`Bad Router ADV Subcode`

is the number of IRDP messages with a bad ICMP subcode.

`Bad Router Words/ADDR`

is the number of IRDP messages with an incorrect address length.

`Errors`

is the number of times an ICMP error was generated. Note that Redirect messages are not included in the total. ICMP errors can be caused by any of the following reasons: invalid IP options, problems in IP packet forwarding, or a UDP server crash.

`Good Routes Recorded`

is the number of valid routes discovered by IRDP messages that have been entered in the TCP/IP route table.

`In Dest Unreachable`

is the number of Destination Unreachable (type 3) messages received.
A Destination Unreachable message is sent to the NonStop TCP/IP subsystem when another host, gateway or router determines that a destination host or port is unreachable. This message can be caused by the following reasons: either there is no route to the destination or the route to the destination has gone down; a nonexistent address has been specified; the process listening on the port has gone down; the destination host has crashed; or fragmentation is needed but the “Don't Fragment” flag is set.

`In Echo`

is the number of Echo (type 8) messages received. The Echo message is sent from the source address to the destination address. An Echo Reply message containing the same data is expected from the destination address.

`In Echo Reply`

is the number of Echo Reply (type 0) messages received. This ICMP message is the reply to the Echo (type 8) message. Essentially, an Echo Reply message is just the original Echo message with the type changed from 8 to 0 and the destination and source addresses reversed; the data returned in the Echo Reply message is the same as that sent in the Echo message. The receipt of an Echo Reply message informs the local host that the remote host is still alive. The data returned also gives the local host a means of testing the integrity of the link.

`In Info Reply`

is the number of Information Reply (type 16) messages received. A host, router, or gateway sends this message—with the source and destination addresses fully specified—in reply to an Information Request message. Note that the Information Request/Reply facility, although supported, is rarely used.

`In Info Request`

is the number of Information Request (type 15) messages received. A host, router, or gateway can send this message—with the network portion of the source address and the destination address set to 0—to determine the number of the network on which it is running. Any host on the network can respond to this request with an Information Reply message.

`In Parameter Problem`

is the number of Parameter Problem (type 12) messages received. A host, router, or gateway sends this message to notify the NonStop TCP/IP subsystem (functioning as a source host) that one of its datagrams has been discarded because the header parameters are incorrect.

`In Redirect`

is the number of Redirect (type 5) messages received. A gateway sends this message to the NonStop TCP/IP subsystem (functioning as a source host) to indicate that there is a shorter path to the destination through another router or gateway.

When the NonStop TCP/IP subsystem receives a Redirect message, it corrects its routing table to reflect the new route. If a host receives many Redirect messages in a short period of time, it is usually an indication that the host is not correcting its routing table.

When the NonStop TCP/IP subsystem services the In Redirect messages, it adds a dynamic route entry of the name #DRT*n*. This dynamic route is used in lieu of the previous route which has been redirected.

`In Source Quench`

is the number of Source Quench (type 4) messages received. A router or gateway sends this message to the NonStop TCP/IP subsystem to indicate it is receiving datagrams more quickly than it can process them.

When the NonStop TCP/IP subsystem receives this message, it reduces the rate at which it is sending datagrams by implementing a slow start. To implement a slow start, the NonStop TCP/IP subsystem first stops sending datagrams, then restarts sending them, and gradually increases the number of datagrams sent.

If the NonStop TCP/IP subsystem is doing a lot of retransmissions, you should check to see if Source Quench messages are being received. If they are, you should reduce the number of packets being transmitted by your applications.

`In Time Exceeded`

is the number of Time Exceeded (type 11) messages received. A router or gateway sends this message to notify the NonStop TCP/IP subsystem (functioning as a source host) that the “time-to-live” field is 0 and that the router or gateway discarded the datagram.
A destination host sends this message if the host cannot reassemble a fragmented datagram within the time limit because fragments are missing. The destination host then discards the datagram. When a Time Exceeded message is received, you should check for routing loops.

`In Timestamp`

is the number of Timestamp (type 13) messages received. A host, router, or gateway sends this message to indicate the last time it handled the message before sending it.

`In Timestamp Reply`

is the number of Timestamp Reply (type 14) messages received. A host, router, or or gateway sends this message in reply to a Timestamp message. This message indicates the time in the original Timestamp message and the time at which the Timestamp message was received by the destination. The Timestamp facility is used to obtain the network time. Special applications can be written to use this facility.

`Invalid Header Size`

is the number of packets received with a length that is shorter than the length specified in the header. This error, usually caused by a noisy link, is rarely reported because the checksum routine also detects this problem.

`Packets Too Short`

is the number of packets received that were shorter than the minimum length allowed for an ICMP packet. Short packets are usually caused by a noisy link.

`Reflect Packets`

is the number of ICMP packets received that have been sent a response. Note that not all ICMP packets require a response.

`Short IP Packets`

is the number of packets received that were too short.

`Out Dest Unreachable`

is the number of Destination Unreachable messages sent.

`Out Echo`

is the number of Echo messages sent.

`Out Echo Reply`

is the number of Echo Reply messages sent.

`Out Info Reply`

is the number of Information Reply messages sent.

`Out Info Request`

is the number of Information Request messages sent.

`Out Parameter Problem`

is the number of Parameter Problem messages sent.

`Out Redirect`

is the number of Redirect messages sent.

`Out Source Quench`

is the number of Source Quench messages sent.

`Out Time Exceeded`

is the number of Time Exceeded messages sent.

`Out Timestamp`

is the number of Timestamp messages sent.

`Out Timestamp Reply`

is the number of Timestamp Reply messages sent.

`Router Advertisement`

is the number of IRDP discovery messages detected by the NonStop TCP/IP subsystem. The NonStop TCP/IP subsystem either records these routes or ignore them, depending on how IRDP is configured and according to route preference.

`Router Solicitation`

is the number of IRDP solicitation messages sent by the NonStop TCP/IP subsystem.

Description of Statistics for QIO

`Current MBUFs Used`

is the current number of MBUFs in use.

`Current Pool Allocation`

is the current number of bytes of pool space in use.

`Data MDs In Use`

is the current number of data MDs in use by the process.

`Dup Driver MDs In Use`

is the current number of duplicate MDs assigned to inbound driver MDs in use by the process.

`Dup MDs in Use`

is the current number of duplicate MDs not assigned to inbound driver MDs in use by the process.

`Maximum Data MDs Used`

is the maximum number of data MDs that have been in use.

`Maximum Dup MDs Used`

is the maximum number of duplicate MDs not assigned to inbound driver MDs that have been in use by the process.

`Max Dup Driv MDs Used`

is the maximum number of duplicate MDs assigned to inbound driver MDs in use by the process.

`Maximum MBUFs Used`

is the maximum number of MBUFs to be used.

`Maximum Pool Allocation`

is the maximum pool space used.

`MBUF Allocation Fails`

is the number of times an MBUF was not available.

`MD Queue Limits`

is the number of times the send or receive queue on a TCP session exceeded a predefined limit of MDs queued. The process attempts to decrease the number queued by collapsing the data into a smaller number of MDs.

`No Data MDs Avail`

is the number of times the process failed to obtain a data MD.

`No Dup Driv MDs Avail`

is the number of times the process failed to obtain a duplicate MD for a driver inbound MD.

`No Dup MDs Avail`

is the number of times the process failed to obtain a duplicate MD.

`Pool Allocation Fails`

is the number of times a pool space request failed.

`QIO Driver Errors`

is the number of times the QIO driver returned an error.

`QIO Limit Warnings`

is the number of times the process received an event signifying a pool or an MD shortage from the QIO monitor.

`Total MBUFs Allocated`

is the current number of MBUFs allocated.

Description of Statistics for Socket Send Size Histogram

`Size 1-128`

is the count of socket sends between 1 and 128 bytes.

`Size 129-256`

is the count of socket sends between 129 and 256 bytes.

`Size 257-512`

is the count of socket sends between 257 and 512 bytes.

`Size 513-1024`

is the count of socket sends between 513 and 1024 bytes.

`Size 1025-2048`

is the count of socket sends between 1025 and 2048 bytes.

`Size 2049-4096`

is the count of socket sends between 2049 and 4096 bytes.

`Size 4097-8192`

is the count of socket sends between 4097 and 8192 bytes.

`Size 8193-12288`

is the count of socket sends between 8193 and 12288 bytes.

`Size 12289-16384`

is the count of socket sends between 12289 and 16384 bytes.

`Size 16385-32768`

is the count of socket sends between 16385 and 32768 bytes.

Description of Statistics for the ARP STATS

`In ARP Requests`

is the number of ARP requests received.

`Out ARP Requests`

is the number of ARP requests sent.

`In ARP Replys`

is the number of ARP replies received.

`Out ARP Replys`

is the number of ARP replies sent.

`In InARP Requests`

is the number of inverse ARP requests received.

`Out InARP Requests`

is the number of inverse ARP requests sent.

`In InARP Replys`

is the number of inverse ARP replies received.

`Out InARP Replys`

is the number of inverse ARP replies sent.

`In ARP Naks`

is the number of ARP Naks received.

`Out ARP Naks`

is the number of ARP Naks sent.

Description of Statistics for IGMP Statistics

`Total Packets Input`

is the total number of IGMP packets received.

`Total Reports Sent`

is the total number of IGMP report packets sent by this process.

`Short Packets`

is the total number of IGMP packets received that were too short.

`Bad Checksum`

is the total number of IGMP packets received that had an incorrect checksum.

`Total Queries Input`

is the total number of IGMP query packets received.

`Bad Queries`

is the total number of IGMP query packets received with the IP destination address not equal to the all hosts group.

`Total Reports Input`

is the total number of IGMP membership reports received.

`Bad Reports`

is the total number of bad IGMP membership reports received.

`Reports For Our Groups`

is the total number of IGMP membership reports received for groups we belong to.

STATS ROUTE Command

The STATS ROUTE command displays the NonStop TCP/IP subsystem statistics for the specified routes.

Note. STATS ROUTE with the RESET option is sensitive.

Command Syntax

```
STATS [ / OUT file-spec / ] [ ROUTE route-spec  ]
      [, RESET ]
```

`OUT` *`file-spec`*

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

`ROUTE` *`route-spec`*

specifies the path on which data is sent in order to reach a destination. The fully-qualified *`route-spec`* has the form:

`$`*`process-name`*`.#`*`route-name`*

If you specify the SCF object type (ROUTE) or any portion of the object name (*`route-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`RESET`

resets the statistical counters to zero. (This option is sensitive.)

Example

The following commands request statistics about all running routes:

```
1-> ASSUME PROCESS $ZTCO
2-> STATS ROUTE.*
```

STATS ROUTE Display Format

The format of the display for a ROUTE object is:

```
TCPIP Stats ROUTE \SYSA.$ZTC0.*

Sample Time ... 23 March 1996, 17:18:25.334
Reset Time .... 23 March 1996, 11:47:47.166

Name                Route Usage
#ROU1                10709D
```

`Sample Time`

is the time when the statistics were sampled (displayed or written to a file).

`Reset Time`

is the time when the counters were last reset to zero.

`Name`

is the name of the route.

`Route Usage`

is the number of times this route was used to send IP datagrams.

STATS SUBNET Command

The STATS SUBNET command displays the NonStop TCP/IP subsystem statistics for the specified SUBNETs. This is a nonsensitive command unless the RESET option is specified.

Command Syntax

```
STATS [ / OUT file-spec / ]  SUBNET subnet-spec
      [ , RESET ]      [ , DETAIL ]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

```
$process-name .#subnet-name
```

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`RESET`

resets the statistical counters to zero.

`DETAIL`

requests the detailed status information for the SUBNET.

Examples

The following example request statistics for all running SUBNETs:

```
1->STATS SUBNET \SYSTEM.$ZTC8.*
```

STATS SUBNET Display Format

The format of the display for the STATS SUBNET command is:

```
TCPIP Stats SUBNET \SYSTEM.$ZTC8.*

Sample Time ... 19 Feb 1998, 9:00:56:.054
Reset time ...  18 Feb 1998, 21:09:10.986

Name      Output   Input   Output   Input   Filter  Filter
          Packets  Packets Errors   Errors  Errors
Timeouts
#LOOP0      0D       0D      0D       0D      0D       OD

SAMPLE TIME ... 19 FEB 1998, 9:00:56:055
RESET TIME .... 19 FEB 1998, 7:36:15.674

NAME      Output   Input   Output   Input   Filter
Filter
          Packets  Packets Errors   Errors  Errors
Timeouts

#SN1         1033D    4496D   0D       0D      0D       0D
```

`Sample Time`

is the time when the statistics were sampled (displayed or written to a file).

`Reset Time`

is the time when the counters were last initialized (set to zero).

`Name`

is the name of the SUBNET.

`Output Packets`

is the number of packets sent by the SUBNET.

`Input Packets`

is the number of packets received by the SUBNET.

`Filter Errors`

indicates the number of errors received from SLSA for filter registrations.

`Filter Timeouts`

indicates that the filter registration is not receiving a reply from SLSA in the allowed time.

`Input Errors`

is the number of errors detected when packets were received by the SUBNET. Each input error also generates one of the following operator messages:

```
DEVICE READ ERROR error ON IOP iopname
DEVICE WRITE ERROR error ON IOP iopname
ERROR error ON IOP iopname
```

`Output Errors`

is the number of errors that occurred when packets were sent by the SUBNET. Each output error also generates one of the following operator messages:

```
DEVICE READ ERROR error ON IOP iopname
DEVICE WRITE ERROR error ON IOP iopname
ERROR error ON IOP iopname
```

Considerations

- The object-name template (wildcard notation) is supported.
- STATS is a nonsensitive command without the RESET option; it is a sensitive command with the RESET option.
- To initialize (set to zero) the statistical counters, use the RESET option. STATS, RESET is sensitive.
- The STATS command returns the time at which the current statistics were sampled and the time at which the counters were last reset.
- If the RESET option is specified, the counters associated with the ADDRMAP object are displayed and then set to 0. The timestamp for the reset operation is recorded and the Last Time Used field is set to 0.

STATUS Command

The STATUS command reports the status of the specified TCP/IP object. The STATUS command with the detail option is only supported for the PROCESS object.

STATUS ENTRY Command

The STATUS ENTRY command displays the dynamic status of the specified entry.

Command Syntax

```
STATUS [  / OUT file-spec /  ] [ ENTRY entry-spec ]
         [ , IPADDRESS  ip-addr ]
         [ , ENDPOINT endpoint-value ]
```

`OUT` *`file-spec`*

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

`ENTRY` *`entry-spec`*

is the name of the specified entry. The fully-qualified *`entry-spec`* has the form:

```
$process-name.#entry-name
```

If you specify the SCF object type (ENTRY) or any portion of the object name (*`entry-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`IPADDRESS`

is the IP address of the specified entry used in cases where an ARP or ATMARP table entry is not named. If you do specify a name, the IPADDRESS attribute is ignored.

`ENDPOINT` *`endpoint-value`*

is the handle being used by the ATM address entry. By specifying the ENDPOINT attribute, you can display all the ATMARP entries that are also using the same ATM address entry.

Examples

The following example returns status information about all entries contained in the ARP, ATMARP, and ATM address tables:

```
1-> ASSUME PROCESS $ZTC0
2-> STATUS ENTRY *
```

STATUS ENTRY Response Display

The format of the STATUS ENTRY display is:

```
TCPIP Status ENTRY \SAMCAT.$ZTC0.*

Name:                  (ARP)    IPADDRESS........
172.16.119.1
  Arp Timer........... 19 (Min) Arp Flags........ (INUSE,COM)
  MacAddress.......... %H00 000C 3920CE

Name: #A2       (ATMARP)       IPADDRESS........ 172.16.192.1
  ATM Arp Timer....... 0 (Min) ATM Arp
Flags...(INUSE,COM,PERM)
  Endpoint............ 0D      ATM State Flags.. (None)
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:A1:00

Name: #PVC       (ATMARP PVC)  IPADDRESS........
172.16.192.200
  ATM Arp Timer....... 20 (Min) ATM Arp
Flags..(INUSE,COM,PERM)
  Endpoint............ 805430096D  ATM State Flags..
(CONN,PVC)
  PVC Name............ #IP.PVC000  Subnet........... #EN1
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:A4:00

Name:            (ATMARP)     IPADDRESS.......
172.16.192.210
  ATM Arp Timer....... 14 (Min)   ATM Arp Flags....
(INUSE,COM)
  Endpoint............ 808613456D ATM State Flags.. (CONN)
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:B2:00

-------------------------------------------------------------
--
Name: ATMentry         (ATM PVC)
  ATM Endpoint Timer.. 0     (Sec)  ATM State Flags.. (OUT)
  Endpoint............ 806359568D   State...........
CONNECTED
  In Use Count........ 1            Subnet........... #EN1
  PVC Name............ #ip.pvc000

Name: ATMentry         (ATM SVC)
  ATM Endpoint Timer.. 803   (Sec)  ATM State Flags.
(OUT,SRVR)
  Endpoint............ 808613456D   State...........
CONNECTED
  In Use Count........ 1            Subnet........... #EN1
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:B2:00
```

`Name`

is the name of the entry. The entry type is indicated in parentheses to the right of the name.

`IPADDRESS`

is the IP address of the entry.

`Arp Timer`

is the time in minutes left to expire.

`Arp Flags`

is the state of the ARP table entry; possible values are: completed, permanent, or in use.

`AtmAddr`

is the ATM address of the destination.

`MacAddress`

is the MAC address of the entry in hexadecimal format.

`ATM Arp Timer`

is the time in minutes left before this ATMARP entry is removed.

`ATM Arp Flags`

is the state of the ATMARP table entry; possible values are: completed, permanent, or in use.

`Endpoint`

is the handle being used by the ATM address entry. By specifying the ENDPOINT attribute, you can display all the ATMARP entries that are also using the same ATM address entry.

`ATM State Flags`

is the state of the ATM connection associated with the ATMARP entry. Possible values are:

- None (NONE).
- Resolving ATM address (RSATM) - ARP entry has issued an ARP request to the ATMARP Server to obtain the ATM address to use.
- Connected (CONN) - the ATM address entry is connected to another ATM endpoint. The ATM connection is ready for data transfer.
- PVC—the ATM address entry is a PVC.

PVC Name

shows the name of the PVC being used on the ATM adapter. Note that the name does not include the name of the ATM adapter and needs to be added in this manner.

Subnet

is the name of the SUBNET to which the PVC is associated.

Considerations

- If the ENDPOINT parameter is supplied, you must issue the command specifying all ENTRY objects, for example: STATUS ENTRY *, ENDPOINT 1234567.
- The IPADDRESS attribute is used in the cases where an ARP or ATMARP table entry is not named. If a name is supplied, the IPADDRESS attribute is ignored.
- By specifying the ENDPOINT attribute, you can display all the ATMARP entries that are also using the same ATM address entry.

STATUS PROCESS Command

The STATUS PROCESS command displays the dynamic state of the TCP/IP process and any ports in use.

Command Syntax

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

OUT *file-spec*

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

PROCESS *process-name*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

DETAIL

requests detailed status information for the process.

Examples

The following examples return status information without and with detail:

```
1->STATUS PROCESS $ZTC0
2->STATUS PROCESS \SYSA.$ZTC0, DETAIL
```

STATUS PROCESS Display Format

The format of the display for the STATUS PROCESS command is:

```
TCPIP Status PROCESS \SYSA.$ZTC0

Status:  STARTED
PPID.................. ( 0,107)      BPID.................. ( 1, 98)

Proto  State       Laddr         Lport    Faddr             Fport  SendQ  RecvQ
 TCP   TIME-WAIT   130.252.12.3  ftp-data 130.252.12.152    11089      0      0
 TCP   TIME-WAIT   130.252.12.3  ftp-data 130.252.12.152    63105      0      0
 TCP   ESTAB       130.252.12.3  ftp      130.252.12.252    57441      0      0
 TCP   TIME-WAIT   130.252.12.3  smtp      130.252.12.8      3309       0      0
```

`Status`

always indicates that the process is STARTED.

`PPID`

is the process ID of the TCP/IP primary process.

`BPID`

is the process ID of the TCP/IP backup process. If TCP/IP is running without a backup process, this field shows (0, 0).

`Proto`

is the protocol associated with the socket, which can be UDP (for a UDP socket), TCP (for a TCP socket), or a protocol number (for a raw IP socket).

`State`

is the current state of the socket; it applies only to sockets whose Proto value is TCP. The possible values are:

`CLOSING`

if waiting for a terminate connection request acknowledgment from the remote site.

`CLOSE-WAIT`

if waiting for a terminate connection request from the local user.

`ESTAB`

if the connection is open and the user can send and receive data. This is the normal state for data transfer.

`FIN-WAIT-1`

if waiting for a terminate connection request from the remote TCP site or if waiting for acknowledgment of the terminate connection request that the process has sent previously.

`FIN-WAIT-2`

if waiting for a termination of data to be received after having sent a FIN (termination of data being sent).

`LISTEN`

if waiting for a connection request from any remote TCP site.

`LAST-ACK`

if waiting for acknowledgment of the terminate connection request previously sent to the remote site (which includes an acknowledgment of its terminate connection request).

`SYN-RCVD`

if waiting for an acknowledgment of a SYN-ACK sent in response to a SYN.

`SYN-SENT`

if waiting for a SYN-ACK after having sent a SYN.

`TIME-WAIT`

if waiting for sufficient time to pass (about two round trips) to be sure that stray packets are flushed from the network.

`Laddr`

is the local Internet address associated with the socket, displayed as four-decimal octets.

`Lport`

is the local port number for either TCP or UDP, depending on the value of Proto. The more common port values are displayed in text form; others are displayed as four-decimal octets.

`Faddr`

is the foreign (remote) Internet address associated with the socket, displayed in four-decimal octets.

`Fport`

is the foreign port number for either TCP or UDP, depending on the value of Proto. The more common port values are displayed in text form; others are displayed as four-decimal octets.

`SendQ`

is the number of bytes of data in the send queue of the socket.

`RecvQ`

is the number of bytes of data in the receive queue of the socket.

EXAMPLES

The following command displays the status of the PROCESS object without the detail option:

```
1-> STATUS PROCESS $ZTC0
```

The following command displays the status of the PROCESS object with the DETAIL option:

```
1-> STATUS PROCESS $ZTC0
```

STATUS PROCESS Display Format With Detail

The format of the display for the STATUS PROCESS command with the detail option is:

```
TCPIP Status PROCESS \SYSA.$ZTC0

Status:  STARTED
PPID.................. ( 0,107)      BPID.................. ( 1, 98)

Proto  State       Laddr         Lport    Faddr            Fport  SendQ  RecvQ
 TCP   LISTEN   0.0.0.0          telnet   0.0.0.0          *      0      0
 TCP   LISTEN   0.0.0.0          ftp      0.0.0.0          *      0      0
 TCP   LISTEN   0.0.0.0          ftp      0.O.O.O          *      0      0
 UDP            0.0.0.0          8000     0.0.0.0          *      0      8128
       ---Multicast Groups---  ---State---
          224.0.0.1               STARTED
          230.17.123.55           STARTING
          239.1.2.3                STOPPED
```

`Status`

always indicates that the process is STARTED.

`PPID`

is the process ID of the TCP/IP primary process.

`BPID`

is the process ID of the TCP/IP backup process. If TCP/IP is running without a backup process, this field shows (0, 0).

`Proto`

is the protocol associated with the socket, which can be UDP (for a UDP socket), TCP (for a TCP socket), or a protocol number (for a raw IP socket).

`State`

is the current state of the socket; it applies only to sockets whose Proto value is TCP. The possible values are:

`CLOSING`

if waiting for a terminate connection request acknowledgment from the remote site.

`CLOSE-WAIT`

if waiting for a terminate connection request from the local user.

`ESTAB`

if the connection is open and the user can send and receive data. This is the normal state for data transfer.

`FIN-WAIT-1`

if waiting for a terminate connection request from the remote TCP site or if waiting for acknowledgment of the terminate connection request that the process has sent previously.

`FIN-WAIT-2`

if waiting for a termination of data to be received after having sent a FIN (termination of data being sent).

`LISTEN`

if waiting for a connection request from any remote TCP site.

`LAST-ACK`

if waiting for acknowledgment of the terminate connection request previously sent to the remote site (which includes an acknowledgment of its terminate connection request).

`SYN-RCVD`

if waiting for an acknowledgment of a SYN-ACK sent in response to a SYN.

`SYN-SENT`

if waiting for a SYN-ACK after having sent a SYN.

`TIME-WAIT`

if waiting for sufficient time to pass (about two round trips) to be sure that stray packets are flushed from the network.

`Laddr`

is the local Internet address associated with the socket, displayed as four-decimal octets.

`Lport`

is the local port number for either TCP or UDP, depending on the value of Proto. The more common port values are displayed in text form; others are displayed as four-decimal octets.

`Faddr`

is the foreign (remote) Internet address associated with the socket, displayed in four-decimal octets.

`Fport`

is the foreign port number for either TCP or UDP, depending on the value of Proto. The more common port values are displayed in text form; others are displayed as four-decimal octets.

`SendQ`

is the number of bytes of data in the send queue of the socket.

`RecvQ`

is the number of bytes of data in the receive queue of the socket.

`Multicast Groups`

indicates the IP multicast group addresses that the TCP/IP connection is listening to.

`Multicast Group States`

is the state of the multicast groups. Possible values are:

`STARTED`

indicates that multicast is operational for the group.

`STARTING`

indicates that the multicast group is transitioning to the STARTED (and operational) state but is not yet fully operational.

`STOPPED`

indicates that multicast is not operational for the group.

STATUS ROUTE Command

The STATUS ROUTE command displays the dynamic status of the specified TCP/IP routes.

Command Syntax

```
STATUS [ / OUT file spec / ] [ ROUTE route-spec ]
```

`OUT` *`file-spec`*

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

`ROUTE` *`route-spec`*

specifies the path on which data is sent in order to reach a destination.The fully-qualified *`route-spec`* has the form:

`$`*`process-name`*`.#`*`route-name`*

If you specify the SCF object type (ROUTE) or any portion of the object name (*`route-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests status information about the specified ROUTE:

```
1-> STATUS ROUTE $ZTC0.#MR2
```

STATUS ROUTE Display Format

The format of the display for the STATUS ROUTE command is:

```
TCPIP Status ROUTE \SYSA.$ZTC0.*

Name                Status         RefCnt

#ROU11               STARTED        0
#ROU9                STARTED        0
#ROU12               STARTED        0
#ROU8                STARTED        1
#ROU3                STARTED        1
```

`Name`

is the name of the route.

`Status`

is the summary state of the route.

`RefCnt`

specifies the number of users currently using the specific route. If the value is greater than zero, an application is currently using the specified route.

STATUS SERVER Command

The SCF STATUS ENTRY command returns details on ATM address entries.

Command Syntax

```
STATUS [ /OUT file-spec/ ] [ SERVER server-spec]
```

OUT *file-spec*

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

SERVER *server-spec*

is the name of the server. The fully-qualified *server-spec* has the form:

$*process-name*.#*server-name*

If you specify the SCF object type (SERVER) or any portion of the object name (*server-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command requests status information on all servers running on the system:

```
1>ASSUME PROCESS $ZTC0
2->STATUS SERVER
```

STATUS SERVER Display Format

The format of the display for the STATUS SERVER command is:

```
TCPIP Status SERVER \SAMCAT.$ZTC0.*

Name: #SRV2
  ATM Endpoint Timer.. 487  (Sec)  ATM State Flags..
(OUT,SRVR)
  Endpoint............ 805582288D  State...........
REGISTERED
  Subnet.............. #EN1
  AtmAddr... (NSAP)
47:00:05:80:FF:E1:00:00:00:F2:1A:29:EB:00:00:00:00:01:B2:00
```

`Name`

is the name of the server.

`ATM Endpoint Timer`

shows the time left before some action may be taken. Under normal circumstances, the time value is the amount of time left before the SUBNET re-registers with the ATMARP server. What happens when the timer expires depends on the state of the entry.

`ATM State Flags`

shows detailed information on the ATM address entry being used by the server. Possible values are:

None (NONE)

Incoming connection (IN)	indicates ATM connection was initiated from the network.
Out going connection (OUT)	indicates ATM connection was initiated by this process.
Resolving IP address (RISP)	ATM address entry has issued an inverse ATM ARP request to obtain the IP address of the other end of the ATM connection.
ATMARP Server Address (SRVR)	indicates an ATM connection to an ATMARP server.

State

indicates the current state for the SERVER object. Possible values are:

FREE	Not is use.
CONNECTING	waiting for an ATM API Connect command to complete.
REGISTERING	Waiting for an ATMARP reply to an ATMARP request to register the SUBNET’s IP address with the ATMARP server.
REGISTERED	The SUBNET has successfully registered with the ATMARP server.

Endpoint

is the handle being used by the ATM address entry.

Subnet

shows the SUBNET to which the server is assigned.

AtmAddr

shows the ATM address of the destination.

STATUS SUBNET Command

The STATUS SUBNET command displays the dynamic state of the TCP/IP SUBNETs.

Command Syntax

```
STATUS [ / OUT file spec / ] [ SUBNET subnet-spec ]
```

OUT *file-spec*

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

SUBNET *subnet-spec*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *subnet-spec* has the form:

$*process-name*.#*subnet-name*

If you specify the SCF object type (SUBNET) or any portion of the object name (*subnet-spec*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following commands request status information about the specified SUBNET:

```
1-> ASSUME PROCESS $ZTC0
2-> STATUS SUBNET #SN2
```

STATUS SUBNET Display Format

The format of the display for the STATUS SUBNET command is:

```
TCPIP Status SUBNET \SYSA.$ZTC0.*

Name               Status
#LOOP0             STARTED
#EN1               STARTED
```

`Name`

is the name of the SUBNET.

`Status`

is the summary state of the SUBNET.

Considerations

- The object-name template (wildcard notation) is supported.
- The summary state of the object does not prevent the STATUS command from being completed successfully.
- The STATUS command does not alter the summary state of the objects.

STOP Command

The STOP command terminates the operation of the specified TCP/IP object. You can stop processes, SUBNETs, and routes. When the operation is complete, the object(s) is in the STOPPED summary state. If the specified objects are in use, the STOP command is not completed. If you attempt to stop an object that is in use or is already in the STOPPED summary state, the NonStop TCP/IP subsystem returns a warning.

This is a sensitive command.

STOP PROCESS Command

The STOP PROCESS command terminates the activity of specified processes in a normal manner. This command stops and deletes the process object and all associated SUBNETs and routes.

Command Syntax

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

OUT *file-spec*

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

PROCESS *process-name*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Examples

The following command terminates the operation of the process $ZTC0:

```
1-> STOP PROCESS $ZTC0
```

STOP ROUTE Command

The STOP ROUTE command terminates the activity of the specified route in a normal manner.

Command Syntax

```
STOP  [ / OUT file-spec / ] [ ROUTE route-spec ]
```

OUT *file-spec*

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

ROUTE *route-spec*

specifies the path on which data is sent in order to reach a destination. The fully-qualified `route-spec` has the form:

```
$process-name.#route-name
```

If you specify the SCF object type (ROUTE) or any portion of the object name (`route-spec`) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command terminates the operation of all routes under the assumed process:

```
1-> ASSUME PROCESS $ZTC0
2-> STOP ROUTE *
```

STOP SUBNET Command

The STOP SUBNET command terminates the activity of the specified SUBNETs in a normal manner.

Command Syntax

```
STOP  [ / OUT file-spec / ] [ SUBNET subnet-spec]
```

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`*`.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

Example

The following command terminates the operation of the SUBNET $ZTC0.#SN1:

```
1-> STOP SUBNET $ZTC0.#SN1
```

Considerations

- The object-name template (wildcard notation) is supported.
- To stop a process immediately, use the ABORT command.

TRACE Command

The TRACE command allows you to capture and store records that you can then display using the PTrace utility. The TRACE command can request the capture of data items, alter trace attributes that were set by a previous use of the command, or stop a previously requested trace operation.

This is a sensitive command.

△ **Caution.** The trace operation can significantly increase processor use by the TCP/IP process. To avoid problems with other processes in the processor, HP recommends that you lower the priority of the TCP/IP process before issuing the TRACE command.

TRACE PROCESS Command

The TRACE PROCESS command traces a process.

Command Syntax

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

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

`OUT` *`file-spec`*

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

`PROCESS` *`process-name`*

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`STOP`

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

`TO`

specifies the name of the file into which the results of the trace operation are to be placed. It is a required option if STOP is not used.

`SELECT`

selects the operations to be traced. For the PROCESS object, you can specify the following for *`select-spec`*:

ALL	All records.
SOCKCMD	Socket requests (bind, listen, accept, connect, send).
MSGSYS	Message system interface.
MALLOC	Resource allocation and deallocation events.
ROUTING	Requests for route changes.
UDP	IDP interface layer.
TCP	Transmission Control Protocol message layer.
IP	IP layer.
LOGIC	Several of the above selections including socket requests (SOCKCMD) and message system interface (MSGSYS).

For more detailed information on the specific trace records captured by each of the SELECT options, including how these records are displayed with the PTrace program, refer to NonStop TCP/IP Trace Facility on page 4-111.

`COUNT`

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

`NOCOLL`

indicates that the trace collector process should not be initiated.

`PAGES`

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

`RECSIZE`

specifies the length, in bytes, of the data in the trace data records. *`size`* is an integer in the range 300 through 4050. The length of the trace header, which is 8 bytes, is not included in *`size`*. If you omit this option or specify 0, an error occurs.

`WRAP`

specifies that when the trace disk file end-of-file (EOF) is reached, trace data wraps around to the beginning of the file and overwrite any data that is there.

Examples

The following command traces the assumed process, writes results into the file named $DATA1.TRC.TRACE, allows the trace data to be overwritten when the EOF is reached, and selects tracing of all TCP/IP process activity:

```
SCF> TRACE PROCESS, TO $DATA1.TRC.TRACE, WRAP, RECSIZE 300, &
        SELECT ALL
```

TRACE SUBNET Command

The TRACE SUBNET command traces a SUBNET.

Command Syntax

```
TRACE [ / OUT file-spec / ] [ SUBNET subnet-spec ]

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

`OUT` *`file-spec`*

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

`SUBNET` *`subnet-spec`*

names the point of connection between the NonStop TCP/IP process and an I/O device. The fully-qualified *`subnet-spec`* has the form:

`$`*`process-name`*`.#`*`subnet-name`*

If you specify the SCF object type (SUBNET) or any portion of the object name (*`subnet-spec`*) in a prior ASSUME command, you can omit it in this command. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`STOP`

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

`TO` *`file-spec`*

specifies the name of the file into which the results of the trace operation are to be placed. It is a required option if STOP is not used.

`SELECT`

selects the operations to be traced.
For the SUBNET object, you can specify the following for *`select-spec`*:

ALL	All records
IPI	IP Input records
IPO	IP Output records
ARPI	ARP Input records
ARPO	ARP Output records
LOGIC	A combination of all the above records
USERDATA	Used with IPI and IPO to display user data

For more detailed information on the specific trace records captured by each of the SELECT options, including how these records are displayed with the PTrace program, refer to NonStop TCP/IP Trace Facility on page 4-111.

`COUNT` *`count`*

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

`NOCOLL`

indicates that the trace collector process should not be initiated.

`PAGES` *`pages`*

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

`RECSIZE`

specifies the length, in bytes, of the data in the trace data records. *`size`* is an integer in the range 300 through 4050. The length of the trace header, which is 8 bytes, is not included in *`size`*. If you omit this option or specify 0, an error occurs.

`WRAP`

specifies that when the trace disk file end-of-file (EOF) is reached, trace data wraps around to the beginning of the file and overwrite any data that is there.

Examples

The following command traces the $ZTC0.#SN2 SUBNET, writes the results into the file $SYSA.TRACES.TCPSUB, allows the trace data to be overwritten when the EOF is reached, and traces all TCP/IP process activity on the SUBNET:

```
SCF> TRACE SUBNET $ZTC0.#SN2, TO $SYSA.TRACES.TCPSUB, WRAP  &
        RECSIZE 300
```

VERSION Command

The VERSION command displays the NonStop TCP/IP subsystem version number, product name, product number, and RVU date. Use the DETAIL option to display this information on the operating system, the SCF Kernel, and the NonStop TCP/IP product module.

This is a nonsensitive command.

VERSION PROCESS Command

Since the null object is supported for VERSION PROCESS, the object PROCESS is optional.

Command Syntax

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

   [ , DETAIL ]
```

`OUT file-spec`

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

`PROCESS process-name`

is a valid process name indicating the desired TCP/IP process. If you omit the object name, SCF uses the assumed object name. For information about the ASSUME command, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for H-Series RVUs.*

`DETAIL`

designates that complete version information is to be returned for the specified object. If DETAIL is omitted, a single line of version information is returned for the object.

Examples

The second example shows the null command.

```
->VERSION PROCESS $ZTC0
->VERSION
->VERSION PROCESS $ZTC0, DETAIL
```

VERSION Command Display Format

The format of the display of the VERSION command without the DETAIL option is:

```
SCF> VERSION PROCESS $ZTC0

VERSION PROCESS \SYSA.$ZTC0: T9551D41^29FEB96^TCPIP^D41002^00
```

VERSION Command Display Format With DETAIL

The format of the display of the VERSION command with the DETAIL option is:

```
SCF> VERSION $ZTC0, DETAIL

Detailed VERSION PROCESS \SYSA.$ZTC0
  SYSTEM \SYSA
    T9551D41^29FEB96^TCPIP^D41002^00
    GUARDIAN - T9050 - (P40)
    SCF KERNEL - T9082F40 - (29FEB96) (01JAN96)
    TCPIP PM - T6243D41 - (29FEB96) - (03SEP96)
```

NonStop TCP/IP Trace Facility

This section contains the following information:

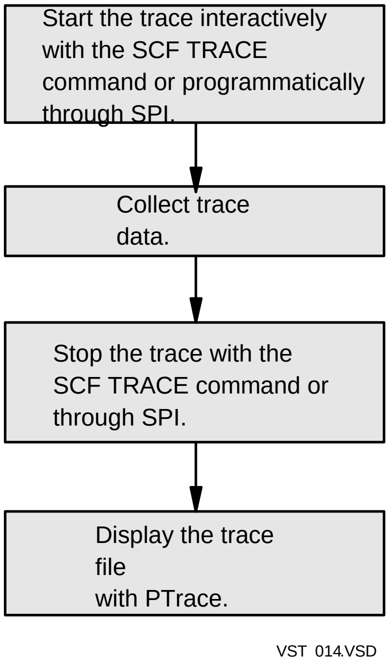
- An introduction to the NonStop TCP/IP trace facility
- A description of the subsystem-specific PTrace commands and any special considerations for using these commands with the NonStop TCP/IP subsystem
- An example of each type of trace record display

Introduction to PTrace

Trace files contain a record of the communications between processes. Each subsystem determines what information is recorded in its trace files. This information varies as to the type of events that are recorded, the amount of detail that is included, and other subsystem-specific attributes.

You can generate a NonStop TCP/IP trace file interactively or programmatically. To start a trace and capture data interactively, you use the SCF TRACE command. To start a trace and capture data programmatically, you use the Subsystem Programmatic Interface (SPI). The trace files created with either SCF or SPI are unstructured and cannot be printed or displayed directly. You use PTrace to display and examine the trace files. The PTrace program formats the data stored in these unstructured trace files for output to terminals, printers, or disk files. Figure 4-2 shows the four general steps involved in recording and formatting trace data.

Figure 4-2. Recording and Displaying Trace Data


Start the trace interactively with the SCF TRACE command or programmatically through SPI.
Collect trace data.
Stop the trace with the SCF TRACE command or through SPI.
Display the trace file with PTrace.
VST 014.VSD

1. Start the trace interactively with the SCF TRACE command or programmatically with SPI.
2. The TRACE command allows you to specify attributes, such as the size of the trace records and the name and maximum size of the trace file.
3. Collect trace data. Send and receive data or perform other operations related to the problem you are analyzing.
4. Stop the trace with another SCF TRACE command or with SPI.
5. Display the trace file with PTrace.

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

Device Type and Subtype

When a trace file is created, the type and subtype of the device being traced are recorded in that file. When PTrace opens the trace file, it uses this information to determine for which subsystem PTrace is formatting records.

The device type and subtype for the NonStop TCP/IP subsystem are 48 and 0, respectively.

PTrace Commands

The PTrace commands provide options for selecting trace records for display, so you can suppress those records that do not relate to the problem you are investigating. The PTrace commands also provide options for specifying the way in which the trace records are formatted.

Although PTrace provides a common set of commands for displaying trace records, not all of the PTrace commands are supported by each subsystem. This is because of the structure of the PTrace code. The PTrace code actually consists of two modules. The first module contains the code shared by all subsystems; the second contains the additional, subsystem-specific code that actually displays the PTrace records. Thus, those commands implemented by the first PTrace module are supported by all subsystems: ALLOW, COUNT, ENV, EXIT, FC, FIND, FROM, HELP, LIMIT, LOG, NEXT, OBEY, OUT, PAGESIZE, RECORD, and RESET. Those additional commands implemented by the subsystem-specific PTrace modules vary from subsystem to subsystem. Of the commands that fall into the subsystem-dependent category, the NonStop TCP/IP subsystem supports the following:

DETAIL

HEX

LABEL

OCTAL

SELECT

TEXT

The HEX, OCTAL, and TEXT commands are implemented in the standard manner. The LABEL and SELECT commands vary slightly from the standard, as described later in this section.

The following subsystem-dependent commands are not supported:

EBCDIC

FILTER

SETTRANSLATE

TEST

TRANSLATE

Table 4-7 lists and describes all the PTrace commands supported by the NonStop TCP/IP subsystem.

Table 4-7. Summary of NonStop TCP/IP PTrace Commands

Command	Description
ALLOW	Specifies the number of errors or warnings permitted during the execution of a command
COUNT	Counts the records in the trace file
DETAIL	Turns on the detailed display formatting of records
ENV	Displays the settings of the PTrace session attributes
EXIT	Terminates a PTrace session
FC	Allows correction of the last PTrace command line entered
FIND	Searches the formatted output for a specified string
FROM	Specifies the trace file to be displayed
HELP	Displays information on TRACE commands
HEX	Sets the hexadecimal display option
LABEL	Turns on subsystem-controlled formatting and display of trace data
LIMIT	Limits the number of records displayed by a single command
LOG	Directs a copy of PTrace input and output to a file
NEXT	Displays the next trace data record(s) in the file
OBEY	Causes commands to be read from a different input file
OCTAL	Sets the octal display option
OUT	Redirects PTrace output
PAGESIZE	Sets the terminal screen size for interactive mode
RECORD	Displays record(s) selected by record number
RESET	Resets session attributes to their default values
SELECT	Establishes selection criteria for displaying records
TEXT	Sets the text display option

The remainder of this subsection describes in detail the subsystem-dependent commands supported by the NonStop TCP/IP subsystem (DETAIL, HEX, LABEL, OCTAL, SELECT, and TEXT).

Each command description includes a brief description of the command, the command's syntax, and any special considerations applicable to the command. The commands are presented in alphabetical order.

See the Notation Conventions on page xviii for a description of the notation scheme used here.

For information on starting PTrace and entering PTrace commands, and for more detailed descriptions of the standard PTrace commands available to all subsystems, refer to the *PTrace Reference Manual*.

DETAIL Command

The DETAIL command controls the detailed display option. When DETAIL is set to ON, PTrace displays extended formatted versions of some records (for example, ARP traffic).

Command Syntax

```
DETAIL [ ON | OFF ]
```

`ON`

turns on detailed display mode.

`OFF`

turns off detailed display mode.

Considerations

- The NonStop TCP/IP DETAIL command is implemented in the standards defined in the *PTrace Reference Manual*.
- If the DETAIL command is not used, the OFF attribute is assumed.
- If DETAIL is specified without the ON or OFF attribute, the ON attribute is assumed.
- The RESET and FROM commands set the DETAIL command to OFF.

HEX Command

The HEX command controls the hexadecimal display option. When HEX is set to ON, PTrace displays a hexadecimal dump of trace-file records (excluding the record header), with character equivalents printed to the right of the dump.

Command Syntax

```
HEX [ ON | OFF ]
```

`ON`

turns on hexadecimal display mode.

`OFF`

turns off hexadecimal display mode.

Considerations

- The NonStop TCP/IP HEX command is implemented in the standards defined in the *PTrace Reference Manual*.
- If the HEX command is not used, the OFF attribute is assumed.
- If HEX is specified without the ON or OFF attribute, the ON attribute is assumed.
- The RESET and FROM commands set the HEX command to OFF.

LABEL Command

The LABEL command controls the formatted display of trace records.

Command Syntax

```
LABEL [ ON | OFF ]
```

`ON`

turns on the formatted display of trace records. The default value is ON.

`OFF`

turns off the formatted display of trace records, but the record header for the trace record is displayed.

Note. The LABEL command is the only way to display NonStop TCP/IP trace records.

Considerations

- If the LABEL command is not used, the ON attribute is assumed.
- If LABEL is specified without the ON or OFF attribute, the ON attribute is assumed.
- The RESET and FROM commands set the LABEL command to ON.

OCTAL Command

The OCTAL command controls the octal display option. When OCTAL is set to ON, PTrace displays an octal dump of trace-file records (excluding the record header), with character equivalents printed to the right of the dump.

Command Syntax

```
OCTAL [ ON | OFF ]
```

`ON`

turns on octal display mode.

`OFF`

turns off octal display mode.

Considerations

- If the OCTAL command is not used, the OFF attribute is assumed.
- If OCTAL is specified without the ON or OFF attribute, the ON attribute is assumed.
- The RESET and FROM commands set the OCTAL command to OFF.

SELECT Command

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

Command Syntax

```
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 a keyword either for the PROCESS object or the SUBNET object.

The following keywords apply to the PROCESS object:

ALL	All records
SOCKCMD	Socket requests (bind, listen, accept, connect, send)
MSGSYS	Message system interface
MALLOC	Resource allocation and deallocation events
ROUTING	Requests for route changes
UDP	IDP interface layer
TCP	Transmission Control Protocol message layer
IP	IP layer
LOGIC	Several of the above selections including socket requests (SOCKCMD) and message system interface (MSGSYS)

The following keywords apply to the SUBNET object:

ALL	All records
IPI	IP input records
IPO	IP output records
ARPI	ARP input records
ARPO	ARP output records
LOGIC	A combination of all the above records
USERDATA	Used with IPI and IPO to display user data

Considerations

- If the SELECT command is not entered, the default mask and keyword is ALL.
- If the SELECT command is specified with no mask or keywords, the ALL keyword is assumed.
- The ENV command allows you to see which SELECT keywords are currently being used.
- PTrace prints the NonStop TCP/IP VPROC version of the NonStop TCP/IP process that is creating a trace file.
- NonStop TCP/IP collects detailed location debugging information with each trace point.
- The ENV command allows you to see which SELECT keywords are currently being used.
- NonStop TCP/IP and PTrace collect and decode detailed SOCKET and internal TCP control block information when SOCKCMD or TCP are selected for the PROCESS object.

Examples

1. The following command sequence can be used to trace and decode SOCKET command requests and responses during a PROCESS object trace:

```
TRACE PROCESS $ZTC0,TO TRACEFL,RECSIZE 750,SELECT SOCKCMD
```

2. The following command sequence can be used to trace and decode ARP protocol traffic during a SUBNET object trace:

```
TRACE SUBNET #E1,TO TRACEFL,RECSIZE 500,SELECT (ARPI,ARPO)
```

3. The following command sequence can be used to trace and decode IP application data (by providing the keyword USERDATA) during a SUBNET object trace:

```
TRACE SUBNET #EN1,TO TRACEFL,RECSIZE 750,        &
   SELECT (IPI,IPO,USERDATA)
```

TEXT Command

The TEXT command controls the text display option. When TEXT is set to ON, PTrace displays an interpreted text of trace-file records (excluding the record header). The textual display appears below labeled data, the HEX display, and the OCTAL display, if they are present. The textual display consists of ASCII characters, with control codes represented by two- or three-character mnemonics.

Command Syntax

```
TEXT [ ON | OFF ]
```

`ON`

turns on text display mode.

`OFF`

turns off text display mode.

Considerations

- The NonStop TCP/IP TEXT command is implemented in the standards defined in the *PTrace Reference Manual*.
- If the TEXT command is not used, the OFF attribute is assumed.
- If TEXT is specified without the ON or OFF attribute, the ON attribute is assumed.
- The RESET and FROM commands set the TEXT command to OFF.

Trace Record Formats

This subsection describes the formatted NonStop TCP/IP trace records. The records are presented in alphabetical order under the SELECT keyword used to display them. The SELECT keyword categories are presented in numeric order, based on their record-type code, as follows:

Type	Record
1	SOCKCR
2	MBUF
3	IPC
4	TCP
5 and 6	UDPI and UDPDI
7	UDPO and UDPDO
9	IPI
10	IPO

Type	Record
11	ROUTE
12	SOCKCMD
13	UDPUREQ

Each description includes the time when the record is generated, the record-type code, the text of the record, and the definitions of any values contained in the record.

Header Format

Each trace record displayed is preceded by a header line having the following format:

```
Date Time timestamp >Delta Time time Record # rec-# Record
Type rec-type
Line line-num of file-name (time on date)
```

line-num of *file-name (time* on *date)*

is the line number that caused the event, the fully-qualified file name, and the last time the file was compiled.

rec-no

indicates the record number. Records are numbered sequentially based on age. The oldest record in the file (the trace file header record) is record 0. The oldest data record is record 1. The newest record in the file is record number *n* - 1, where *n* is the number of records in the file. The letter D following the record number indicates that it is in double-integer format.

rec-type

indicates the type of trace record.

seq-no

indicates the sequence number. The sequence number is included to keep track of records that are lost when the trace file is written to disk. The sequence number counts from 0 to 255 and then begins again.

time

indicates the time since the last trace run on this line.

timestamp

indicates the timestamp of the record. The timestamp reports the time at which the record was captured. The resolution is to one hundredth of a second.

type

indicates the record-type code. The record-type code identifies the type of information contained in the record. It is subsystem-dependent.

Socket Creation Records

This subsection describes the formatted trace records displayed when the SOCKCR keyword is specified for the PTrace SELECT command. Note that all of the socket creation records are preceded by a header containing the record-type code 1. The records are presented in alphabetical order, based on the text format.

Attach Socket Protocol Record

The attach socket protocol record is generated when a socket is attached to a protocol.

```
header
attach socket_handle nnnnaaaa proto #n
```

nnnnaaaa

indicates the internal socket ID of the socket being attached to a protocol.

n

indicates the IP number of the protocol being attached to the socket. For a list of commonly used IP numbers, refer to the *TCP/IP Programming Manual*. For a complete list of the IP numbers, refer to Request for Comments document 1010, “Assigned Numbers.”

Soclose Record

The soclose record is generated each time the SOCLOSE procedure is called. The SOCLOSE procedure completes the close of a socket.

```
header

procedure:soclose socket_handle nnnnaaaa
```

line-num of *file-name* (*time* on *date*)

is the edit-line number that caused the event, the fully-qualified edit-file name, and the last time the edit file was compiled.

nnnnaaaa

indicates the internal socket ID of the socket being closed.

Sofree Record

The sofree record is generated each time the SOFREE procedure is called. The SOFREE procedure frees up a socket data structure.

```
header

procedure:sofree freeing socket_handle nnnaaa
```

nnnnaaaa

indicates the internal ID of the socket being freed.

Socket Closing Record

The socket closing record is generated when the process initiates the close of a socket.

```
header

socket closing socket_handle nnnnaaaa
```

nnnnaaaa

indicates the internal ID of the socket being closed.

Allocating PCB Record

The allocating PCB record is generated each time a protocol control block (PCB) is allocated for a TCP socket.

```
header

socket_handle nnnnaaaa allocating PCB for TCP socket
```

nnnnaaaa

indicates the internal ID of the socket for which the PCB is being allocated.

Can’t Create New TCPCB Record

The can't create new TCPCB record is generated each time the NonStop TCP/IP process can't create a new control block for a TCP socket.

```
header

socket_handle nnnnaaaa can't create new tcpcb for TCP socket
```

nnnnaaaa

indicates the internal ID of the socket for which the new control block could not be created.

Creating New TCPCB Record

The creating new TCPCB record is generated each time a new control block is created for a TCP socket.

```
header

socket_handle nnnnaaaa creating new tcpcb for TCP socket
```

nnnnaaaa

indicates the internal ID of the socket for which the new control block is being created.

Reserve Space Record

The reserve space record is generated each time a socket structure needs to be created for an incoming TCP connection.

```
header

socket_handle nnnnaaaa reserve space for incoming connection
```

nnnnaaaa

indicates the internal ID of the socket being reserved.

Memory Buffer Allocation Records

This subsection describes the formatted trace records displayed when the MBUF keyword is specified for the PTrace SELECT command. Note that there is only one memory buffer allocation record and that each memory buffer allocation record in the trace file is preceded by a header containing the record-type code 2.

Memory Buffer Allocation Record

The memory buffer allocation record is generated each time the NonStop TCP/IP process attempts to allocate memory buffers (MBUFs). Note that this record is generated even when the allocation attempt fails.

```
header

m_mbufalloc nnnnnnnnnn bytes result: rrrrrrr
```

nnnnnnnnnn

indicates the number of bytes allocated.

rrrrrrr

indicates whether the memory allocation attempt succeeded or not. The value can be *succeed* or *failed*.

Interprocess Communication Records

This subsection describes the formatted trace records displayed when the IPC keyword is specified for the PTrace SELECT command. Note that there is only one interprocess communication record and that each socket system call record in the trace file is preceded by a header containing the record-type code 3.

Socket System Call Record

The socket system call record is generated each time a socket call is made by an application-level program.

```
header

socket sys call #c socket_handle nnnnaaaa received bbb bytes
```

c

indicates the socket call number being invoked. The socket call number is described in the SYSCALH INCLUDE file.

nnnnaaaa

indicates the internal ID of the socket to which the system call applies.

bbb

indicates the number of bytes of data received in the socket call.

TCP Records

This subsection describes the formatted trace records displayed when the TCP keyword is specified for the PTrace SELECT command. Note that TCP records are preceded by a header containing the record-type code 4. The records are presented in alphabetical order, based on their text format.

Data Acked Record

The data acked record is generated each time an ACK is received for the local socket.

```
header

socket_handle nnnnaaaa: acked ack-bytes,
sb_cc unack_bytes
```

nnnnaaaa

indicates the internal socket ID.

ack-bytes

indicates the number of bytes of data acknowledged.

unack-bytes

indicates the number of bytes of data in the queue waiting to be acknowledged.

All Data Acked Record

The all data acked record is generated each time all of the data in the queue has been acknowledged.

```
header

socket_handle nnnnaaaa: acked ack-bytes >
sb_cc unack-bytes, all data acked
```

nnnnaaaa

indicates the internal socket ID.

ack-bytes

indicates the number of bytes of data acknowledged.

unack-bytes

indicates the number of bytes of data in the queue waiting to be acknowledged. Because the number of bytes acknowledged is greater than this value (>), all of the data in the queue has been acknowledged.

After Changes Record

The after changes record is generated each time data or an ACK is received for a TCP socket. Note that the values reported indicate the values of these variables after they have been updated by the packet. The preliminary values are reported in the send next record.

```
header

socket_handle nnnnaaaa: After Changes: snd_nxt snd-nxt,
snd_una snd-una, snd_max snd-max
```

nnnnaaaa

indicates the internal socket ID.

snd-nxt

indicates the next sequence number to be sent.

snd-una

indicates the oldest unacknowledged sequence number.

snd-max

indicates the maximum sequence number that can be sent.

Send Next Record

The send next record is generated each time data or an ACK is received for a TCP socket. Note that the values reported indicate the values of these variables before they have been updated by the packet. The updated values are reported in the after changes record.

```
header

socket_handle nnnnaaaa: snd_nxt snd-nxt, snd_una snd-una
ti_ack ti-ack
```

nnnnaaaa

indicates the internal socket ID.

snd-nxt

indicates the next sequence number to be sent.

snd-una

indicates the oldest unacknowledged sequence number.

ti-ack

indicates the sequence number of the data currently being acknowledged.

Receive State Change Record

The receive send state change record is generated when data is received.

```
header

socket_handle nnnnaaaa tcp_handle nnnnn
init-state: input (start-no..end-no) @ ack-no,
urp=urp [f1,f2,f3,f4,f5,f6] -> fin-state...
rcv_(nxt,wnd,up) (rcv-nxt, rcv-wnd, rcv-up)
snd_(una,nxt,max) (snd-una, snd-nxt, snd-max)
snd_(wl1,wl2,wnd) (snd-wl1, snd-wl2, snd-wnd)
```

nnnnaaaa

indicates the internal socket ID.

nnnnn

indicates the internal ID of the TCP packet.

init-state

indicates the initial state before the data was received. The possible states are:

CLOSE-WAIT
CLOSED
CLOSING
ESTABLISHED
FIN-WAIT-1
FIN-WAIT-2
LAST-ACK
LISTEN
SYN-RECVD
SYN-SENT
TIME-WAIT

start-no

indicates the starting sequence number of the data received.

end-no

indicates the ending sequence number of the data received.

ack-no

indicates the acknowledgment number.

urp

indicates the urgent pointer.

[*f1,f2,f3,f4,f5,f6*]

indicates the control flags set. The possible flags that can be set are SYN, ACK, FIN, RST, PUSH, and URG.

fin-state

indicates the final state after the data was received. The possible states are:

- CLOSE-WAIT
- LAST-ACK
- CLOSED
- LISTEN
- CLOSING
- SYN-RECVD
- ESTABLISHED
- SYN-SENT
- FIN-WAIT-1
- TIME-WAIT
- FIN-WAIT-2

rcv-nxt

indicates the next sequence number expected to be received.

rcv-wnd

indicates the receive window.

rcv-up

indicates the receive urgent pointer.

snd-una

indicates the oldest unacknowledged sequence number.

snd-nxt

indicates the next sequence number to be sent.

snd-max

indicates the maximum sequence number that can be sent.

snd-wl1

indicates the sequence number used for the last window update.

snd-wl2

indicates the acknowledgment number used for the last window update.

snd-wnd

indicates the send window.

Send State Change Record

The send state change record is generated when a user sends data.

```
header

socket_handle nnnnaaaa tcp_handle nnnnn
init-state: user req-type -> fin-state...
rcv_(nxt,wnd,up) (rcv-nxt, rcv-wnd, rcv-up)
snd_(una,nxt,max) (snd-una, snd-nxt, snd-max)
snd_(wl1,wl2,wnd) (snd-wl1, snd-wl2, snd-wnd)
```

nnnnaaaa

indicates the internal socket ID.

nnnnn

indicates the internal ID of the TCP packet.

init-state

indicates the initial state before the data was sent. The possible states are:

- CLOSE-WAIT
- LAST-ACK
- CLOSED
- LISTEN
- CLOSING
- SYN-RECVD
- ESTABLISHED
- SYN-SENT
- FIN-WAIT-1
- TIME-WAIT
- FIN-WAIT-2

req-type

indicates the request type. The possible request types are:

- ABORT
- PEERADDR
- ACCEPT
- PROTORCV
- ATTACH
- PROTOSEND
- BIND
- RCVD

- CONNECT
- RCVOOB
- CONNECT2
- SEND
- CONTROL
- SENDOOB
- DETACH
- SENSE
- DISCONNECT
- SHUTDOWN
- FASTIMO
- SLOWTIMO
- LISTEN
- SOCKADDR

`fin-state`

indicates the final state after the data was sent. The possible states are:

- CLOSE-WAIT
- LAST-ACK
- CLOSED
- LISTEN
- CLOSING
- SYN-RECVD
- ESTABLISHED
- SYN-SENT
- FIN-WAIT-1
- TIME-WAIT
- FIN-WAIT-2

`rcv-nxt`

indicates the next sequence number expected to be received.

`rcv-wnd`

indicates the receive window.

`rcv-up`

indicates the receive urgent pointer.

`snd-una`

indicates the oldest unacknowledged sequence number.

`snd-nxt`

indicates the next sequence number to be sent.

snd-max

indicates the maximum sequence number that can be sent.

snd-wl1

indicates the sequence number used for the last window update.

snd-wl2

indicates the acknowledgment number used for the last window update.

snd-wnd

indicates the send window.

Accepting Connection Record

The accepting connection record is generated each time an incoming connection is accepted on a local socket.

```
header
socket_handle nnnnaaaa: tcp_usrreq: PRU_ACCEPT
   faddr forgn-addr fport forgn-port
```

nnnnaaaa

indicates the internal socket ID.

forgn-addr

indicates the remote Internet address associated with the incoming connection.

forgn-port

indicates the remote port number associated with the incoming connection.

Incoming Connection Record

The incoming connection record is generated each time an incoming connection request is received on a local socket.

```
header
socket_handle nnnnaaaa: tcp_usrreq: PRU_CONNIND
   faddr forgn-addr fport forgn-port
```

nnnnaaaa

indicates the internal socket ID.

forgn-addr

indicates the remote Internet address associated with the incoming connection.

forgn-port

indicates the remote port number associated with the incoming connection.

TCP Socket Request Record

The TCP socket request record is generated each time a TCP socket request is made.

```
header
socket_handle nnnnaaaa: tcp_usrreq: socket request #nnnnn
```

nnnnaaaa

indicates the internal socket ID.

nnnnn

indicates the internal request number used to manipulate the TCP socket. The possible values that can appear and their meanings are discussed in the PROTOSWH INCLUDE file.

UDP Input Records

This subsection describes the formatted trace records displayed when the UDPI keyword is specified for the PTrace SELECT command. Note that UDP input records are preceded by a header containing the record-type code 5 or 6. The records are presented in alphabetical order, based on their text format.

Received UDP Packet Record

The received UDP packet record is generated each time the UDP input routine is executed. This record is preceded by a header containing the record-type code 6.

```
header
Received UDP packet for udp_header_handle nnnnaaaa
```

nnnnaaaa

indicates the internal ID of the UDP packet.

Sent UDP Packet to User Record

The sent UDP packet to user record is generated each time a valid user is identified for an incoming UDP packet and the packet is delivered to the user. This record is preceded by a header containing the record-type code 5.

```
header
udp_input: Sent UDP packet to user --> udp_header_handle
nnnnaaaa
```

nnnnaaaa

indicates the internal ID of the UDP packet.

Detailed UDP Input Records

This subsection describes the formatted trace records displayed when the UDPDI keyword is specified for the PTrace SELECT command. Note that detailed UDP input records are preceded by a header containing the record-type code 5 or 6. The records are presented in alphabetical order, based on their text format.

Destination Address and Port Record

The destination address and port record is generated each time a UDP packet is received. This record is preceded by a header containing the record-type code 5.

```
header
udp_input: dst dst-addr, dport port-no
udp_header_handle nnnnaaaa
```

dest-addr

indicates the packet's destination Internet address.

port-no

indicates the packet's destination UDP port number.

nnnnaaaa

indicates the internal ID of the UDP packet.

Packet Length Record

The packet length record is generated each time a UDP packet is received. This record is preceded by a header containing the record-type code 6.

```
header
udp_input: packetlen lllll udp_header_handle nnnnaaaa
```

lllll

indicates the packet's length.

nnnnaaaa

indicates the internal ID of the UDP packet.

Source Address and Port Record

The source address and port record is generated each time a UDP packet is received. This record is preceded by a header containing the record-type code 5.

```
header
udp_input: src ip-addr, sport portno
udp_header_handle nnnnaaaa
```

ip-addr

indicates the packet's source Internet address.

portno

indicates the packet's source UDP port number.

nnnnaaaa

indicates the internal ID of the UDP packet.

UDP Output Records

This subsection describes the formatted trace records displayed when either the UDPO or UDPDO keyword is specified for the PTrace SELECT command. Note that UDP output records are preceded by a header containing the record-type code 7. The records are presented in alphabetical order, based on their text format.

UDP Sending From Record

The UDP sending from record is generated each time the NonStop TCP/IP process sends a packet.

```
header
udp_output: sending from  ip-addr.udp-port
```

ip-addr

indicates the source Internet address.

udp-port

indicates the source UDP port number.

UDP Sending to Record

The UDP sending to record is generated each time the NonStop TCP/IP process sends a packet.

```
header
udp_output: sending to ip-addr.udp-port
```

ip-addr

indicates the destination IP address.

udp-port

indicates the destination UDP port number.

IP Input Records

This subsection describes the formatted trace records displayed when the IPI keyword is specified for the PTrace SELECT command. Note that IP input records are preceded by a header containing the record-type code 9. The records are presented in alphabetical order, based on their text format.

Sending ICMP Error Record

The sending ICMP error record is generated each time the IP detects an error and requests the generation of an ICMP error packet.

```
header
ip_forward: ip_handle nnnnaaaa sending icmp error
dst 1234cccc, code ptype
```

nnnnaaaa

indicates the internal ID of the IP packet.

1234cccc

indicates the destination Internet address in the packet containing the error.

ptype

indicates the type of ICMP packet requested. The NonStop TCP/IP subsystem supports the following packet types and packet-type codes:

```
Echo Reply (0)
Destination Unreachable (3)
Source Quench (4)
Redirect (5)
Echo (8)
Time Exceeded (11)
```

```
w Problem (12)
Timestamp (13)
Timestamp Reply (14)
Information Request (15)
Information Reply (16)
```

Forwarding to IP Address Record

The forwarding to IP address record is generated each time the IP input routines receive a packet destined for another destination.

```
header
ipintr: ip_handle nnnnaaaa forwarding to ip address ip-addr
```

nnnnaaaa

indicates the internal ID of the IP packet.

ip-addr

indicates the address to which the packet is forwarded.

Got Fragment Record

The got fragment record is generated each time the IP input routines receive a packet fragment.

```
header
ipintr: ip_handle nnnnaaaa got fragment offset bbbbb
```

nnnnaaaa

indicates the internal ID of the IP packet.

bbbbb

indicates the IP offset (in bytes).

Packet for Us Record

The packet for us record is generated each time the IP input routines receive a packet destined for this address.

```
header
ipintr: ip_handle nnnnaaaa packet for us, proto #nnnnn
```

nnnnaaaa

indicates the internal ID of the IP packet.

nnnnn

indicates the IP protocol number (either 6 for TCP or 17 for UDP).

Rebuilt Fragment Record

The rebuilt fragment record is generated each time the IP input routines rebuild a packet from packet fragments.

```
header
ipintr: ip_handle nnnnaaaa rebuilt fragment len lllll
```

nnnnaaaa

indicates the internal ID of the IP packet.

lllll

indicates the rebuilt packet's total length.

Message Buffer Length Record

The message buffer length record is generated each time the IP input routines are executed.

```
header
ipintr: mbuflen lllll
```

lllll

indicates the length of the packet received.

IP Output Records

This subsection describes the formatted trace records displayed when the IPO keyword is specified for the PTrace SELECT command. Note that IP output records are preceded by a header containing the record-type code 10. The records are presented in alphabetical order, based on their text format.

Destination IP Address Record

The destination IP address record is generated each time the IP sends a packet with a standard destination address.

```
header
ip_output: dest ip address ip-addr, proto ppppp
```

ip-addr

indicates the destination IP address.

ppppp

indicates the IP number associated with the packet sent. For a list of the commonly used IP numbers, refer to the *TCP/IP Programming Manual.* For a complete list of the IP numbers, refer to Request for Comments document 1010, “Assigned Numbers.”

Fragmenting Record

The fragmenting record is generated each time the IP must fragment a packet.

```
header
ip_output: fragmenting offset bbbbb
```

bbbbb

indicates the IP offset of the fragments (in bytes).

Sending Broadcast Record

The sending broadcast record is generated each time the IP sends a packet with a broadcast address.

```
header
ip_output: sending broadcast len lllll
```

lllll

indicates the length of the broadcast packet sent.

Route Records

This subsection describes the formatted trace records displayed when the ROUTE keyword is specified for the PTrace SELECT command. Note that route records are preceded by a header containing the record-type code 11. The records are presented in alphabetical order, based on their text format.

Flags Record

The flags record is generated each time a route change request is received.

```
header
flags ffff
```

ffff

indicates the internal flags set during the routing change. The value displayed represents a bit pattern in which bit 0 is the low-order bit and bits 1 through 5 correspond to the following flags: bit 1 indicates whether the route is UP, bit 2 indicates whether the route is to a gateway or router, bit 3 indicates whether the route is to a point-to-point connection, bit 4 indicates whether the route is marked down, and bit 5 indicates whether the route is a dynamic route.

Route Addition Record

The route addition record is generated each time a route is added. Note that this record does not return any values.

```
header
req SIOCADDRT
```

Route Deletion Record

The route deletion record is generated each time a route is deleted. Note that this record does not return any values.

```
header
req SIOCDELRT
```

Route Request Record

The route request record is generated each time a route change request is received.

```
header
rtreq: dst dst-addr, gateway gw-addr
```

dst-addr

indicates the destination address associated with the route change request.

gw-addr

indicates the gateway/router address associated with the route change request.

Socket Command Records

This subsection describes the formatted trace records displayed when the SOCKCMD keyword is specified for the PTrace SELECT command. Note that socket command records are preceded by a header containing the record-type code 12. The records are presented in alphabetical order, based on their text format.

Accept Record

The accept record is generated each time a connection is accepted on the local socket.

```
header
accept: socket_handle nnnnaaaa connection on
1234abcd.12345
```

nnnnaaaa

indicates the internal socket ID.

1234abcd.12345

indicates the remote IP address and port number.

Address Family Record

The address family record is generated each time a connection request is received on the local socket.

```
header
AF fffff
```

fffff

indicates the address family for the new connection.

Bind Record

The bind record is generated each time a name (consisting of a local Internet address and port number) is bound to a socket.

```
header
bind: socket_handle nnnnaaaa, port ppppp, local_addr
loc-addr
```

nnnnaaaa

indicates the internal socket ID.

ppppp

indicates the local port number to be associated with the socket.

loc-addr

indicates the local Internet address to be associated with the socket.

Connection Request Record

The connection request record is generated each time a connection request is received on the local socket.

```
header
connect: socket_handle nnnnaaaa, to address
1234abcd.12345
```

nnnnaaaa

indicates the internal socket ID.

1234abcd.12345

indicates the remote IP address and port number.

Connection Waiting Record

The connection waiting record is generated each time the socket has to wait for a connection to complete.

```
header
connect: waiting socket_handle nnnnaaaa
```

nnnnaaaa

indicates the internal socket ID.

Queue Length Record

The queue length record is generated when a listen call is made.

```
header
listen: socket_handle nnnnaaaa qlen lllll
```

nnnnaaaa

indicates the internal socket ID.

lllll

indicates the maximum queue length of pending TCP connections on the socket.

Waiting for Reply Record

The waiting for reply record is generated each time an accept call is not completed immediately (that is, if the socket has to wait for an incoming connection).

```
header
listen: socket_handle nnnnaaaa waiting for reply
```

nnnnaaaa

indicates the internal socket ID.

Send Record

The send record is generated each time a send call is made.

```
header
send: socket_handle nnnnaaaa bbbbb
```

nnnnaaaa

indicates the internal socket ID.

bbbbb

is number of bytes transferred.

Send to Record

The send to record is generated each time a sendto call is made.

```
header
sendto: socket_handle nnnnaaaa bbbbb, to address
1234abcd.1234
```

nnnnaaaa

indicates the internal socket ID.

bbbbb

is number of bytes transferred.

1234abcd,12345

indicates the remote IP address and port number.

Socket Family Record

The socket family record is generated each time a socket is created.

```
header
sock_reply: family fffff, type ttttt, proto proto
```

fffff

indicates the address family specified by the programmer in the socket call.

ttttt

indicates the socket type specified by the programmer in the socket call.

proto

indicates the IP number specified by the programmer in the socket call (either 0 for IP, 6 for TCP, or 17 for UDP).

Socket Reply Record

The socket reply record is generated each time a socket request is completed.

```
header
sock_reply: socket_handle nnnnaaaa
```

nnnnaaaa

indicates the internal socket ID.

UDP User Request Records

This subsection describes the formatted trace records displayed when the UDPUREQ keyword is specified for the PTrace SELECT command. Note that UDP user request records are preceded by a header containing the record-type code 13. The records are presented in alphabetical order, based on their text format.

Socket Request Record

The socket request record is generated each time a UDP socket request is made.

```
header
udp_usrreq: socket_handle nnnnaaaa, socket request #nnnnn
```

nnnnaaaa

indicates the internal socket ID.

nnnnn

indicates the internal request number used to manipulate the UDP socket. The possible values that can appear and their meanings are explained in the PROTOSWH INCLUDE file.

UDP Socket Request Completed Record

The UDP socket request completed record is generated each time a UDP socket request is completed with an error.

```
header
udp_usrreq: socket request nnnnn completed with
error err-no
```

nnnnn

indicates the internal request number associated with the UDP socket request. The possible values that can appear and their meanings are explained in the PROTOSWH INCLUDE file.

err-no

indicates the error code returned as the result of the socket request. For descriptions of the error codes returned, refer to the *TCP/IP Programming Manual*.

A Configuration Reference

This appendix provides reference material required for configuring the NonStop TCP/IP subsystem.

Use the following list to access the material you need quickly:

Other NonStop TCP/IP Services

Many components make up the NonStop TCP/IP subsystem. This subsection describes additional NonStop TCP/IP services that are included as part of the NonStop TCP/IP software library. The descriptions that follow specifically point out any nonconformance to the protocols and standards that is required by the NonStop operating system.

The programmatic interface provided by the socket library is described in detail in the *TCP/IP Programming Manual*.

Domain Name Server (DNS)

The Domain Name Server (DNS) serves as the yellow pages and white pages of an internet community. It provides the translation and mapping of human-readable machine names into IP addresses. Host table lookup routines (such as the Domain Name Resolver, described below) work well for small networks that include only a few machines where the organizations in the network cooperate in maintaining the tables: a master file for the entire network can be maintained at a central location by a few people. However, a single centralized clearinghouse for host-name resolution does not work well for large networks which include machines that cross organizational boundaries. The DNS allows the authority for this information to be delegated to the organizations on the network that are responsible for it.

The DNS, as defined in RFCs 1032, 1033, 1034, and 1035, allows a network to be divided into a hierarchy of domains. The name space is organized as a tree structure according to organizational or administrative boundaries. Each node (called a domain) is given a label, and the name of the domain is derived by concatenating all the labels of the domains from the root of the tree down to the current domain. A label need only be unique within its domain.

The whole name space is partitioned into several areas called zones, each starting at a domain and extending down to the leaf domains or to domains where other zones start. Zones usually represent administrative boundaries. The following example shows an address for a host at KentComm:

```
guru.develop.KentComm.com
```

com is the top level domain for commercial organizations. KentComm is a subdomain of com, and develop is a subdomain of KentComm. The host itself, guru, belongs to the domain develop.

The program file that implements the DNS is $SYSTEM.ZTCPIP.NAMED, so the DNS is sometimes referred to as NAMED. The basic function of the DNS is to provide information about network objects by answering queries. The information provided by the DNS includes name-to-address mapping, address-to-name mapping, mailbox information, and machine hardware/software information.

For more detailed information on domain names, refer to RFC 1034 and RFC 1035 available from the DDN Network Information Center. Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994*.*

The Domain Name Server (DNS) was ported from BSD BIND 4.8.

Note. You should read RFCs 819, 920, 974, 1032, 1033, 1034, 1035, and 1101 before you attempt to configure the server.

Types of Domain Name Servers

There are four basic types of Domain Name Servers:

- Master
- Caching-Only
- Remote
- Slave

Master Server

A master server for a domain is the authority for that domain. This server maintains all the data corresponding to its domain. Each domain should have at least two master servers: a primary master and one (or more) secondary masters to provide backup service if the primary is unavailable or overloaded. A server may be a master for multiple domains, primary for some domains, and secondary for others. The functions of each type of server are:

Primary Server	A primary master server loads its data from a disk file. This server also can delegate authority to other servers in its domain.
Secondary Server	A secondary master server is delegated authority and receives its data from a primary master server. At startup time, a secondary server requests all the data for the given zone from the primary master server. The secondary server periodically checks with the primary server to see if the primary needs to update its data.

Caching-Only Server

All servers are caching servers, meaning that the server caches the information it receives for use until the data expires. A caching-only server is a server that is not authoritative for any domain. This server services queries and asks other servers that are authoritative for the information needed. All servers keep data in their cache until the data expires. The expiration time is based on a time-to-live field (the time after which the data in the cache becomes invalid) associated with the data when it is received from another server.

Remote Server

A remote server is an option given to people who would like to use a domain name server on their workstation or on a machine that has a limited amount of memory and CPU cycles. By using this option you can run all the networking programs that use the domain name server without having to run the domain name server on the local machine. All queries are serviced by a domain name server running on another machine on the network.

Slave Server

A slave server is a server that always forwards to a fixed list of forwarding servers those queries it cannot satisfy locally, instead of interacting with the master name servers for the root and other domains. The queries to the forwarding servers are recursive queries. There may be one or more forwarding servers, and they are tried in turn until the list is exhausted.

A slave and forwarder configuration is typically used when you do not want all the servers at a given site to interact with the rest of the Internet servers. A typical scenario would involve a number of workstations and a departmental timesharing machine that has Internet access. The workstations might be administratively prohibited from having Internet access. To give the workstations the appearance of access to the Internet domain system, the workstations could be slave servers to the timesharing machine, which forwards queries and interacts with other name servers to resolve the query before returning the answer.

An added benefit of using the forwarding feature is that the central machine develops a much more complete cache of information that all the workstations can use. The use of slave mode and forwarding is discussed in more detail under the description of the domain name server's startup commands in Domain Name Server Files on page A-5.

Domain Name Resolver

The Domain Name Resolver is part of the NonStop TCP/IP sockets library; Domain Name Resolver allows user-developed application programs to access a Domain Name Server, as described above. The Domain Name Resolver consists of those socket library support routines that get information on hosts, networks, protocols, and services.

Depending on which support routine your program calls and the setting of the =TCPIP^HOST^FILE parameter at the time the program is executed, the Domain Name resolver either accesses a name server or a special file that contains a list of Internet addresses with the host name and aliases that correspond to each address. The default name of this file is $SYSTEM.ZTCPIP.HOSTS. If this information is contained in some other file, each user running the program must set the =TCPIP^HOST^FILE parameter to specify the name of that file.

If a name server is available on the network, the recommended method for resolving names is to access the name server. To ensure that the resolver accesses a name server, your program should call gethostbyname or gethostbyaddr, and you should not set the TCPIP^HOST^FILE parameter before running the program. For more information on gethostbyname or gethostbyaddr, see the *TCP/IP Programming Manual*.

The resolver uses information specified in a configuration file to provide access to a name server. The default name for this file is $SYSTEM.ZTCPIP.RESCONF. When an application sends a `name_resolution` request to the resolver, the resolver sends the request to the servers listed in the RESCONF file in order of their priority in a timed sequence. The server listed first in the RESCONF file (the primary server) has the highest priority. The RESCONF file must contain a minimum of one server; the maximum is three servers. For more information on setting up the RESCONF file, refer to RESCONF File on page 3-36.

Domain Name Server Files

The domain name server uses several files to load its database. This subsection describes the files and the formats needed for NAMED.

NAMED listens to TCP and UDP port 53 for incoming requests. To change the Default NAMED port number, use the -p option; however, make sure that the port you have chosen is not in use by another service or application. To start NAMED on a port number less than 1024, you must be in the SUPER group.

The -b option specifies the boot file. The default boot file is $SYSTEM.ZTCPIP.DNSBOOT.

Lines in the files are divided into fields. If a field in the line is optional, the notation [*optional-field-value*] is used. If the field is mandatory, the notation *req-field* is used.

If a field must be used as specified, the field is shown as it must be used (for example, IN means that the string IN should appear in the line in the position shown).

Boot File

The boot file is first read when NAMED starts up. This file informs the server what type of server it is, which zones it has authority over, and where to get its initial data. The default file is $SYSTEM.ZTCPIP.DNSBOOT. However, you can specify the name of a different boot file on the startup command line.

The boot file consists of lines of the following format:

```
field-name argument ....
```

The various field names and the arguments are described below.

Domain

You may specify a default domain for the name server by entering a line such as the following:

```
domain zone-name
```

For example:

```
domain  kentcomm.com
```

The name server uses this information when it receives a query for a name that has no period (.) and is unknown. When the name server receives such a query, it appends the *zone-name* to the query name.

Assume that the following name is received in a query:

```
xyz
```

The result of applying the default domain is:

```
xyz.kentcomm.com
```

Primary Master

The line in the boot file that designates the server as a primary server for a zone has the following format:

```
primary  zone-name  data-file-name
```

For example:

```
primary  kentcomm.com  $SYSTEM.ZTCPIP.DNSHOSTS
```

This record indicates that the server is a primary server for *zonename*, and *data-file-name* contains data for this *zone-name.*

Primary master also should have a record for the IN-ADDR.ARPA domain of the format:

```
primary  internet_address.in-addr.arpa  data-file-name
```

For example:

```
primary  32.128.in-addr.arpa  $SYSTEM.ZTCPIP.DNSREV
```

For an explanation of IN-ADDR.ARPA domain, refer to Domain Data Files, later in this section.

Secondary Master

The line for a secondary server is similar to the line for a primary master, except that it lists addresses of other servers (usually primary servers) from which the zone data is obtained. The format of the line is as follows:

```
secondary  zone-name primary1 [primary2...] [backup-file]
```

For example:

```
secondary  BigCityU.Edu  128.33.0.11 $SYSTEM.ZTCPIP.DNSBAK
```

The first field specifies that the server is a secondary master server for the zone stated in the second field. The network address specifies the domain name server that is primary for the zone. The secondary server gets its data across the network from the listed server.

The secondary master also should have a record for the IN-ADDR.ARPA domain of the format:

```
secondary  internet_address.in-addr.arpa  data-file-name
```

For example:

```
secondary  32.128.in-addr.arpa  $SYSTEM.ZTCPIP.DNSREV
```

For an explanation of IN-ADDR.ARPA domain, refer to Domain Data Files, later in this section.

More than one network address may be listed here (separated by spaces). Each server is tried in the order listed until it successfully receives the data from a listed server. If you specify a backup file, data for the zone is dumped into that file as a backup. There is no default backup file.

Caching-Only Server

A special line is not required to designate that a server is a caching server. Instead, a caching-only server is revealed by the absence of authority lines, such as secondary or primary in the boot file.

All servers must have a line in the boot file to prime the domain name server’s cache:

```
cache               .        root-cache-file-name
```

For example:

```
cache               .        $SYSTEM.ZTCPIP.NAMEDCA
```

Note that the period (.) between the keyword cache and the *root-cache-file-name* is required. There is no default cache file.

All cache files listed are read in at the NAMED startup (boot) time. Any values still valid are reinstated in the cache, and the root name server information in the cache files is always used. For information on cache files refer to Cache Initialization, later in this section.

Forwarders

Forwarders are the catch-all name servers. Any server can use forwarders. A forwarder is another server capable of processing recursive queries and willing to try to resolve queries on behalf of other systems. The forwarder command specifies forwarders by Internet address, as follows:

```
forwarder [internet-addr [internet-addr...]]
```

For example:

```
forwarder 128.33.0.12 128.33.0.14
```

There are two reasons for using forwarders:

- The other systems may not have full network access and may be prevented from sending any IP packets into the rest of the network; such systems must rely on a forwarder that does have access to the full net.
- The forwarder sees a union of all queries as they pass through the server; therefore, the forwarder builds up a rich cache of data compared to the cache in a typical workstation name server. In effect, the forwarder becomes a super-cache that all hosts can benefit from, thereby reducing the total number of queries from that site to the rest of the network.

Slave Mode

Slave mode is used if forwarders provide the only possible way to resolve queries due to lack of full net access, or if you wish to prevent the name server from using something other than the listed forwarders. Slave mode is activated by entering the following command:

```
slave
```

Enter this command in the boot file. If you use a slave, you must specify forwarders. When in slave mode, the server forwards each query to each forwarder until it finds an answer or exhausts the list of forwarders.

Remote Server

To set up a host that uses a remote server instead of a local server to answer queries, set up the file $SYSTEM.ZTCPIP.RESCONF as specified in RESCONF File on page 3-36. This file designates which domain name servers on the network should be sent queries. Do not create (set up) this file if you have a local server running, since if this file exists, it is read almost every time gethostbyname or gethostbyaddr is called programmatically.

Cache Initialization

The domain name server needs to know the servers that are the authoritative domain name servers for the root domain of the network. To provide this information to the domain name server, you must prime the domain name server's cache with the addresses of these higher authorities. The location of the file containing the cache entries is specified in the boot file. This cache file uses the Standard Resource Record Format (also known as Masterfile Format).

Domain Data Files

Any of four standard files can specify the data for a domain: DNSCACHE, DNSLOCAL, DNSHOSTS, DNSREV. These files use the Standard Resource Record Format.

DNSCACHE The name server needs to know the server that is the authoritative name server for the network. You must prime the name server’s cache with the address of higher authorities. Specify the location of this initialization file in the boot file.

DNSLOCAL This file specifies the address for the local loopback interface, better known as local host (or me). The network address is 127.0.0.1. Specify the location of this file in the boot file.

DNSHOSTS This file contains all the data about the machines in this zone. Specify the location of this file in the boot file.

DNSREV This file specifies the IN-ADDR.ARPA domain. This special domain allows address-to-name mapping (reverse of name-to-address mapping). Because Internet host addresses do not fall within domain boundaries, this domain was formed to allow inverse mapping. The IN-ADDR.ARPA domain has four labels preceding it, which correspond to the four octets of an Internet address. You must specify all four octets even if an octet’s value is zero. The Internet address 128.33.0.14 is located in the domain 14.0.33.128.IN-ADDR.ARPA. This reversal of the address is awkward to read, but allows for the natural grouping of hosts in a network.

Example Boot Files

The following section contains sample boot files for the different types of servers. A boot file is the startup file that specifies the mode of operation and specifies other information to NAMED. The boot file is named $SYSTEM.ZTCPIP.DNSBOOT

Primary Master Server

In this example of a boot file, the domain name server is set to act as a primary master server:

```
;
; Boot file for Primary Master Name Server
;
; type      domain                 source file or host
;
domain      kentcomm.com
primary     kentcomm.com           $SYSTEM.ZTCPIP.DNSHOSTS
primary     33.128.in-addr.arpa    $SYSTEM.ZTCPIP.DNSREV
primary     0.0.127.in-addr.arpa   $SYSTEM.ZTCPIP.DNSLOCAL
cache       .                      $SYSTEM.ZTCPIP.DNSCACHE
```

The domain specified is kentcomm.com. The domain name server that uses this boot file is a primary for this domain and a primary for reverse-address mapping in the special domain IN-ADDR.ARPA. The domain name server is to load its cache from the file $SYSTEM.ZTCPIP.DNSCACHE.

Secondary Master Server

In this example of a boot file, a domain name server is set to act as a secondary master server:

```
;
; Boot file for Secondary Master Name Server
;
; type     domain              source file or host
domain     kentcomm.com
secondary  kentcomm.com        128.33.0.14 $SYSTEM.ZTCPIP.DNSBAK
secondary  33.128.in-addr.arpa 128.33.0.11
$SYSTEM.ZTCPIP.DNREVBAK
primary   0.0.127.in-addr.arpa $SYSTEM.ZTCPIP.DNSLOCAL
cache     .                    $SYSTEM.ZTCPIP.DNSCACHE
```

In this example, you may specify more than one address on the secondary line. Use spaces to separate the addresses. The domain name server that uses this example boot-file is the secondary for the domain kentcomm.com. The server downloads its database from the address 128.33.0.14 and writes this information to the file $SYSTEM.ZTCPIP.DNSBAK.

If the secondary server cannot access the primary domain-name server’s database at boot time, it loads its database from the file $SYSTEM.ZTCPIP.DNSBAK (if the file exists). When the secondary server can contact the primary, it checks to see if the primary has a different serial number (see the SOA—Start Of Authority on page A-16).

If the primary has a different serial number, new information is loaded from the primary, and the DNSBAK file is updated.

This domain name server is the secondary server for the reverse-address mapping of the domain 33.128.IN-ADDR.ARPA. The information required for this server is loaded from the file $SYSTEM.ZTCPIP.DNREVBAK. The version in the SOA record is compared with the primary server specified in the address 128.33.0.11. The data is written to the file $SYSTEM.ZTCPIP.DNREVBAK if the version is older than that specified by the primary domain name server.

If the secondary server cannot access the primary domain-name server's database at boot time (startup time), the information loaded from the file $SYSTEM.ZTCPIP.DNREVBAK (if it exists) is used. When the secondary server can contact the primary, it checks to see if the primary has a different serial number (see the SOA—Start Of Authority on page A-16). If the primary has a different serial number, new information is loaded from the primary and the DNREVBAK file is updated.

This domain name server is the primary server for the reverse-address mapping in the special domain 0.0.127.INADDR.ARPA. The cache for the domain name server is to be loaded from $SYSTEM.ZTCPIP.DNSCACHE.

Caching-Only Server

In this example of a boot file, a domain name server is set to act as a caching-only server:

```
; Boot file for Caching-Only Name Server
;
; type      domain                 source file or host
;

domain      kentcomm.com
cache       .                      $SYSTEM.ZTCPIP.DNSCACHE
primary     0.0.127.in-addr.arpa   $SYSTEM.ZTCPIP.DNSLOCAL
```

In this example, the database cache is loaded from $SYSTEM.ZTCPIP.DNSCACHE. This server is the primary server for reverse-address mapping in the local special domain 0.0.127.IN-ADDR.ARPA.

Remote Server

To access a remote server, set up the file $SYSTEM.ZTCPIP.RESCONF as specified earlier in RESCONF File on page 3-36.

Example Data Files

The following paragraphs contain sample domain database files for the domain name server.

DNSCACHE

The DNSCACHE file primes the address of the root-domain name servers, basically giving the domain name server a location to start searches.

An example of the DNSCACHE file is:

```
; Initial cache data for root domain servers.
;
.                      99999999   IN   NS   SRI-NIC.ARPA.
                       99999999   IN   NS   NS.NASA.GOV.
                       99999999   IN   NS   TERP.UMD.EDU.
                       99999999   IN   NS   A.ISI.EDU.
                       99999999   IN   NS   BRL-AOS.ARPA.
                       99999999   IN   NS   GUNTER-ADAM.ARPA.
                       99999999   IN   NS   C.NYSER.NET.

;  Prep the cache (hotwire the addresses).
SRI-NIC.ARPA.          99999999   IN   A    10.0.0.51
SRI-NIC.ARPA.          99999999   IN   A    26.0.0.73
NS.NASA.GOV.           99999999   IN   A    128.102.16.10
A.ISI.EDU.             99999999   IN   A    26.3.0.103
BRL-AOS.ARPA.          99999999   IN   A    128.20.1.2
BRL-AOS.ARPA.          99999999   IN   A    192.5.25.82
BRL-AOS.ARPA.          99999999   IN   A    192.5.22.82
GUNTER-ADAM.ARPA.      99999999   IN   A    26.1.0.13
C.NYSER.NET.           99999999   IN   A    128.213.5.17
TERP.UMD.EDU.          99999999   IN   A    10.1.0.17
```

In the example, the 99999999 is used as a very large number in the time-to-live field. This number indicates that the data will be valid for 99999999 seconds (several years).

The example specifies seven known root domain name servers. These servers are to be queried in order to resolve names.

The second part of the file specifies the Internet address of these domain name servers. This specification is necessary so NAMED does not have to resolve the address of these root-name servers.

DNSLOCAL

An example of the DNSLOCAL file is as follows:

```
@  IN   SOA   jiffy.kentcomm.com. joe.jiffy.kentcomm.com. (
        1.1     ; Serial Number
        3600    ; Refresh Time
        300     ; Retry Time
        3600000 ; Expire Time
        3600 )  ; Minimum Time
   IN   NS    pubs.kentcomm.com.
1  IN   PTR   local host.
```

In this example, a domain name server is on jiffy.kentcomm.com. The maintainer of this database is joe.jiffy.kentcomm.com. The serial number of this database is 1.1. This serial number must increment every time this file is altered.

DNSHOSTS

The DNSHOSTS file specifies the name-to-address mapping along with other information on the hosts. This file is not the same as the hosts file ($SYSTEM.ZTCPIP.HOSTS).

An example of the DNSHOSTS file is as follows:

```
@          IN    SOA     jiffy.kentcomm.com.
joe.jiffy.kentcomm.com. (
                         1.1       ; Serial
                         3600      ; Refresh
                         300       ; Retry
                         3600000   ; Expire
                         3600 )    ; Minimum
           IN    NS      pubs.kentcomm.com.
           IN    NS      jiffy.kentcomm.com.
local host IN    A       127.1
jiffy      IN    A       128.33.4
           IN    A       10.0.0.78
           ANY   HINFO   SUN-SPARCSTATION-20 UNIX
lightning  IN    CNAME   jiffy
larry      IN    A       128.33.0.6
           ANY   HINFO SUN-SPARCSTATION-20 UNIX
stooge1    IN    CNAME   larry
curly      IN    A       128.33.0.7 stooge2
           IN    A       128.33.130.6
           ANY   HINFO HP-9000-750 HPUX
           IN    WKS     128.33.130.6 UDP tftp timed
           IN    WKS     128.33.130.6 TCP ( echo telnet
                         ftp finger smtp )
stooge2    IN    CNAME   curly
pubs       IN    A       10.2.0.78
           IN    A       128.33.0.11
           ANY   HINFO   TANDEM/K2000 GUARDIAN
           IN    WKS     128.33.0.11 TCP ( echo telnet
                         ftp finger smtp )
fred       IN    MX      0  jiffy.kentcomm.com
joe        ANY   MB      tardy.KCI.COM
postmaster ANY   MR      joe
Bind       ANY   MINFO   Bind-Request kjd.BigCityU.Edu.
           ANY   MG      Ralph.BigCityU.Edu.
           ANY   MG      Zhou.BigCityU.Edu.
           ANY   MG      Painter.BigCityU.Edu.
           ANY   MG      Riggle.BigCityU.Edu.
           ANY   MG      Terry.pa.XYZCorp.Com.
```

In the example, the MG and MINFO lines do not have much meaning in the NonStop environment, because they are typically integrated with mail systems. On NonStop machines the mail system is maintained by TRANSFER, which cannot have the domain name server integrated with it. Do not include any MG or MINFO records for NonStop hosts.

In this example, the machine curly has two addresses (128.32.0.7 and 128.32.130.6). The machine type is an HP 9000 Model 750, running HPUX that has timed and TFTP services over UDP, and ECHO, TELNET, FTP, FINGER, and SMTP services over TCP.

These entries should be refreshed every hour (3600 seconds). They expire in 1000 hours (3600000 seconds). This refresh time and expiration time is given to the other systems (such as secondary domain name servers). Secondary domain name servers use this information to periodically update their databases.

DNSREV (Reverse Address-to-Name Mapping)

The DNSREV file shows the file responsible for the address-to-name mapping.

An example of the DNSREV file is:

```
;
;

@       IN   SOA   jiffy.kentcomm.com. joe.tardy.KCI.COM. (
                   1.1     ; Serial
                   3600    ; Refresh
                   300     ; Retry
                   3600000 ; Expire
                   3600 )  ; Minimum
        IN   NS    pubs.kentcomm.com.
        IN   NS    jiffy.kentcomm.com
6.0     IN   PTR   fred.kentcomm.com.
11.0    IN   PTR   jiffy.kentcomm.com.
14.0    IN   PTR   pubs.kentcomm.com.
6.130   IN   PTR   wilma.kentcomm.com.
```

Assume in the example, this is the database for the domain 33.128.IN-ADDR.ARPA. Here, the address 128.33.0.14 would translate to pubs.kentcomm.com. The address 128.33.130.6 would translate to wilma.kentcomm.com.

Standard Resource Record Format

The records in the domain-name server data files are called resource records. The Standard Resource Record Format is specified in RFC 882 and RFC 973. Also, see *DNS and BIND* by Paul Albitz and Cricket Liu for a more information about this topic. Resource records have a standard format, as follows:

[*name*] [*ttl*] *addr-class record-type record-specific-data*

The first field *name* is always the name of the domain record, and it always starts in column 1. If you omit *name*, the resource record takes on the name of the previous resource record.

The second field *ttl* is an optional time-to-live field. This field specifies how long (in seconds) this data should be stored-in the by the resolver. By leaving this field blank, the default time-to-live is specified in the Start of Authority resource record (see below).

The third field `addr-class` is the address class. There are currently two address classes: IN for Internet addresses and ANY for all address classes.

The fourth field `record-type` states the type of the resource record.

The record-specific fields `record-specific-data` are dependent on the type of the resource record.

All comparisons and lookups in the domain name server database are case insensitive.

The following characters have special meanings:

. A free-standing dot in the name field refers to the current domain.

@ A free-standing @ (at-sign) in the name field denotes the current origin.

.. Two free-standing dots represent the null domain name of the root when used in the name field.

X A backslash preceding the character *X*, which can be any character other than a digit (0- 9), prevents its special meaning from being applied. For example, “\”. can be used to place a dot character in a label.

DDD A backslash preceding the decimal number described by *DDD*, where each *D* is a digit, is the octet corresponding to the number. The resulting octet is assumed to be text and is not checked for special meaning.

( ) Parentheses group data that crosses a line. In effect, line terminations are not recognized within parentheses.

; A semicolon starts a comment; the remainder of the line is ignored.

* An asterisk signifies a wildcard.

Most resource records have the current origin appended to names if they are not terminated by a period (.). This feature is useful for appending the current domain name to the data, such as machine names, but may cause problems where you do not want this to happen. If the name is not in the domain for which you are creating the data file, a good rule of thumb is to end the name with a period (.).

$INCLUDE

An include line begins with $INCLUDE, starting in column 1, and is followed by a file name. This feature is particularly useful for separating different types of data into multiple files. For example:

```
$INCLUDE $SYSTEM.ZTCPIP.MBOX
```

The line is interpreted as a request to load the file $SYSTEM.ZTCPIP.MBOX. The $INCLUDE command does not cause data to be loaded into a different zone or tree. The command allows data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

$ORIGIN

You can use the origin line to change the origin in a data file. The line starts in column 1 and is followed by a domain origin. Use this line to put more than one domain in a data file. For example:

```
$ORIGIN  kentcomm.com
```

SOA—Start Of Authority

The format of this record is:

```
name [ttl] addr-class SOA origin person-in-charge (
        serial-number
        refresh-time
        retry-time
        expire-time
        minimum-time )
```

For example:

```
  @ IN SOA   pubs.kentcomm.com joe.pubs.kentcomm.com. (
          1.1       ; Serial Number
          3600      ; Refresh Time
          300       ; Retry Time
          3600000   ; Expire Time
          3600 )    ; Minimum Time
```

The *Start of Authority* (SOA) record designates the start of a zone.

The *name* is the name of the zone.

The *origin* is the name of the host on which this data file resides.

The *person-in-charge* is the mailing address for the person responsible for the domain name server.

The *serial-number* is the version number of this data file. This number must be incremented whenever a change is made to the data. The domain name server cannot handle numbers over 9999 after the decimal point.

The *refresh-time* indicates how often, in seconds, a secondary domain name server is to check with the primary domain name server to see if an update is needed.

The *retry-time* indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh.

The *expire-time* is the upper limit, in seconds, that a secondary domain name server is to use the data before it expires because it did not get a refresh.

The *minimum-time* is the default number of seconds to be used for the time-to-live field on resource records. Each zone should have only one SOA record. The data that follows a semicolon is ignored and treated as comments. The maximum value between the time-to-live in the RR and the MINIMUM in SOA is sent in the response.

NS—Name Server

The format of this record is:

[*name*] [*ttl*] *addr-class* NS *name-server-name*

For example:

```
             IN         NS  jiffy.kentcomm.com
```

In the example, note that the *name* and the *ttl* fields are empty.

The Name Server record (NS) lists a domain name server responsible for a given domain.

The first field *name* lists the domain that is serviced by the listed domain name server. Each Primary Master server for the domain should have one NS record. Use an NS RR to specify the name server at the top of the current domain and the servers child domain. In the latter case, also specify a glue address RR (refer to RFC 1034).

A—Address

The format of this record is:

[*name*] [*ttl*] *addr-class* A *address*

For example:

```
eric         IN         A   128.33.0.14
             IN         A   11.0.0.87
```

The Address record (A) lists the address for a given machine.

The *name* is the machine name and the *address* is the network address. There should be one A record for each address of the machine. If the *name* field is empty, the name is assumed to be the last name specified in previous lines. When the A RR refers to a name server, it is also referred to as glue (refer to RFC 1034).

HINFO—Host Information

The format of this record is:

[*name*] [*ttl*] *addr-class* HINFO *hardware oper-sys*

For example:

```
sam          ANY        HINFO TANDEM/VLX  GUARDIAN
```

In the example, note that the *ttl* field is absent. When you omit the *name* field, the name is assumed to be the last name field that was seen. The Host Information resource record (HINFO) is for host-specific data. This record lists the hardware and operating system that are running at the listed host. Note that only a single space separates *hardware* and *oper-sys* fields. If you want to include a space in the machine name, you must enclose the name in quotation marks. Host information is not specific to any address class, so you can use ANY for the address class. Each host should have one HINFO record.

WKS—Well-Known Services

The format of this record is:

[*name*] [*ttl*] *addr-class* WKS *address protocol services*

For example:

```
pubs         IN         WKS  128.33.0.11  UDP who route timed
             IN         WKS  128.33.0.11  TCP  (echo telnet
                                               discard sunrpc
                                               sftp uucp-path
                                               systat
                                               daytime)
```

The Well-Known Service record (WKS) describes the well-known services supported by a particular protocol at a specified address. The list of services and port numbers comes from the list of services specified in the $SYSTEM.ZTCPIP.SERVICES file. Use only one WKS record for each protocol per address.

CNAME—Canonical Name

The format of this record is:

aliases [*ttl*] *addr-class* CNAME *canonical-name*

For example:

```
bcumonet      IN         CNAME  monet
```

The Canonical Name resource record (CNAME) specifies an alias for a canonical name. An alias should be the only record associated with the alias name; all other resource records should be associated with the canonical name, and not with the alias. Any resource records that include a domain name as their value (such as NS or MX) should list the canonical name, not the alias.

PTR—Domain Name Pointer

The format of this record is:

name [*ttl*] *addr-class* PTR *ireal-name*

For example:

```
    7.0  IN         PTR  monet.BigCityU.Edu.
```

A Domain Name Pointer record (PTR) allows special names to point to some other location in the domain. In the example, a PTR record sets up reverse pointers for the special IN-ADDR.ARPA domain. PTR names should be unique to the zone.

MB—Mailbox

The format of this record is:

name [*ttl*] *addr-class* MB *machine*

For example:

```
jim        IN         MB  jiffy.kentcomm.com.
```

A Mailbox record (MB) lists the machine where a user wants to receive mail. The name field is the user's logon password (or correspondent name in KentComm Guardian machines); the machine field denotes the machine to which mail is to be delivered. Mail Box names should be unique to the zone.

MR—Mail Rename Name

The format of this record is:

name [*ttl*] *addr-class* MR *corresponding-MB*

For example:

```
Postmaster IN         MR  jim
```

The Mail Rename record (MR) can list aliases for a user. The name field lists the alias for the name listed in the fourth field corresponding-MB, which should have a corresponding MB record.

MINFO—Mailbox Information

The format of this record is:

name [ttl] *addr-class* MINFO *requests maintainer*

For example:

```
BIND       IN         MINFO  BIND-REQUEST kjd.BigCityU.Edu.
```

The Mail Information record (MINFO) creates a mail group for a mailing list. This resource record usually is associated with a mail group Mail Group, but may be used with a mail box record.

The *name* field specifies the name of the mailbox. The *requests* field specifies where mail, such as requests to be added to a mail group, should be sent. The *maintainer* field is a mailbox that receives error messages. This mailbox is particularly appropriate for mailing lists when errors in member's names are reported to a person other than the sender.

MG—Mail Group Member

The format of this record is:

mail-group-name [*ttl*] *addr-class* MG *member-name*

For example:

```
                      IN         MG Bloom
```

The Mail Group (MG) lists the members of a mail group.

An example for setting up a mail list is as follows:

```
Bind      IN   MINFO   Bind-Request kjd.BigCityU.Edu.
          IN   MG      Ralph.BigCityU.Edu.
          IN   MG      Zhou.BigCityU.Edu.
          IN   MG      Painter.BigCityU.Edu.
          IN   MG      Riggle.BigCityU.Edu.
          IN   MG      Terry.pa.XYZCorp.Com.
```

MX—Mail Exchanger

The format of this record is:

name [ttl] *addr-class* MX *preference-value mailer-exchanger*

For example:

```
Miller.OY.AB. IN      MX  0  Seismo.ESS.GOV.
*.IN.         IN      MX  0  RELAY.ES.NET.
```

The Mail Exchanger (MX) record specifies a machine that knows how to deliver mail to a machine that is not directly connected to the network. In the first example, Seismo.ESS.GOV. is a mail gateway that knows how to deliver mail to Miller.OY.AB., but other machines on the network cannot deliver mail directly to Miller. These two machines may have a private connection or use a different transport medium. The *preference-value* is the order that a mailer should follow when more than one way exists to deliver mail to a single machine. Refer to RFC 974 for more detailed information on preference values.

You may use wildcard names containing an asterisk (*) for mail routing through MX records. The network is likely to include servers which simply state that any mail sent to a domain is to be routed through a relay. In the second example, all mail to hosts in the domain *.IN is routed through RELAY.ES.NET. This routing is accomplished by creating a wildcard resource record, which states that *.IN has an MX of RELAY.ES.NET.

B

NonStop TCP/IP Processes and Protocols

This appendix provides additional information about the NonStop implementation of Transmission Control Protocol/ Internet Protocol. It also includes information on the services provided by the NonStop TCP/IP environment not documented in Section 1, Configuration Quick Start and Section 3, Configuring the NonStop TCP/IP Subsystem.

Note. This appendix is not intended to replace the reference books listed in About This Manual.

TCP is a set of rules for controlling data transmission; IP is a set of rules for passing data in an internetworked environment. The TCP/IP protocols originated from research funded by the Defense Department in the late 1960s in interconnecting networks that used different hardware. The result was the DARPA (Defense Advanced Research Projects Agency) Internet, now commonly known as the Internet.

Internet Concepts and Services

The basic concept of internetworking is that cooperating networks use common protocols to make the entire structure appear as a collection of networks to any device that has access to it. The Internet continues to grow exponentially in the United States and abroad.

The Internet provides a number of different services using a variety of protocols, in addition to the TCP protocol. The services and protocols available in the NonStop implementation of TCP/IP are described in the following paragraphs.

Addressing

Every transport endpoint, or socket, can be addressed by a combination that includes an Internet address (IP address) and a port number. This section describes Internet addresses and port numbers, both of which are used in configuration and programming.

Internet Addresses

A host can have more than one Internet address because it can be connected to more than one network. Each physical device, or interface, that adds the host to a new logical network has its own Internet address.

A host has only one Internet address on each network to which it is attached. However, because of the multiprocessing nature of the NonStop system and the fact that more than one NonStop TCP/IP process can exist on a system, a NonStop system can appear to the outside world as more than one logical host on a network. When a system appears as more than one logical host, a one-to-one correspondence between a logical host on a network and its Internet address on that network still exists. Multiple NonStop TCP/IP processes in a system on the same IP network must be represented by multiple Internet addresses, so that they appear to the IP network as multiple hosts.

An Internet address is a four-octet (32-bit) numeric value identifying a particular network (network address portion) and a local host on that network (local address portion), as defined in RFC 1010. The standard external representation of an Internet address—known as dotted decimal format—is the ASCII value of each of the four octets, separated by periods (for instance, 133.50.85.43). Each octet value is a number in the range 0 to 255 inclusive, expressed either in decimal with no prefix or in hexadecimal preceded by the characters 0X (with X in either upper-case or lower-case). For example, two other ways to express the address just given are all hexadecimal (0x85.0x32.0x55.0x2b), and a combination of decimal (ASCII) and hexadecimal (133.0X32.85.0X2B). Example B-1 shows how a 32 bit address translates into dotted decimal format.

Example B-1. Dotted Decimal Format for IP Addresses

```
The 32 bit Internet address:

10000010 00001010 000001110 00011110

is written in the more readable decimal form as:

130.10.6.30
```

Network Order and Host Order

When describing some support routines in the socket library, this manual (and others in the NonStop TCP/IP library) refers to Internet addresses or port numbers as being in network order or in host order. These terms refer to the order in which the octets are stored in arguments passed to or returned by the routines.

The Internet standard for transmission of 32-bit integers specifies that the most-significant octet appears first. However, not all hosts store integers in the same way. Thus, the direct copying of octets from one host to another can change the value of a number. The standard specifies that sending hosts must translate from their local integer representation (local order) to network order (most-significant octet first). Receiving hosts are required to translate from network order to local order.

On NonStop systems, the network order is the same as the host order.

Internet Address Classes

If you want to connect a LAN or a host through a router or gateway to any network in the Internet, you must apply to have a range of Internet addresses assigned to you by the Network Information Center (NIC) operated by SRI International. The NIC’s mailing address is:

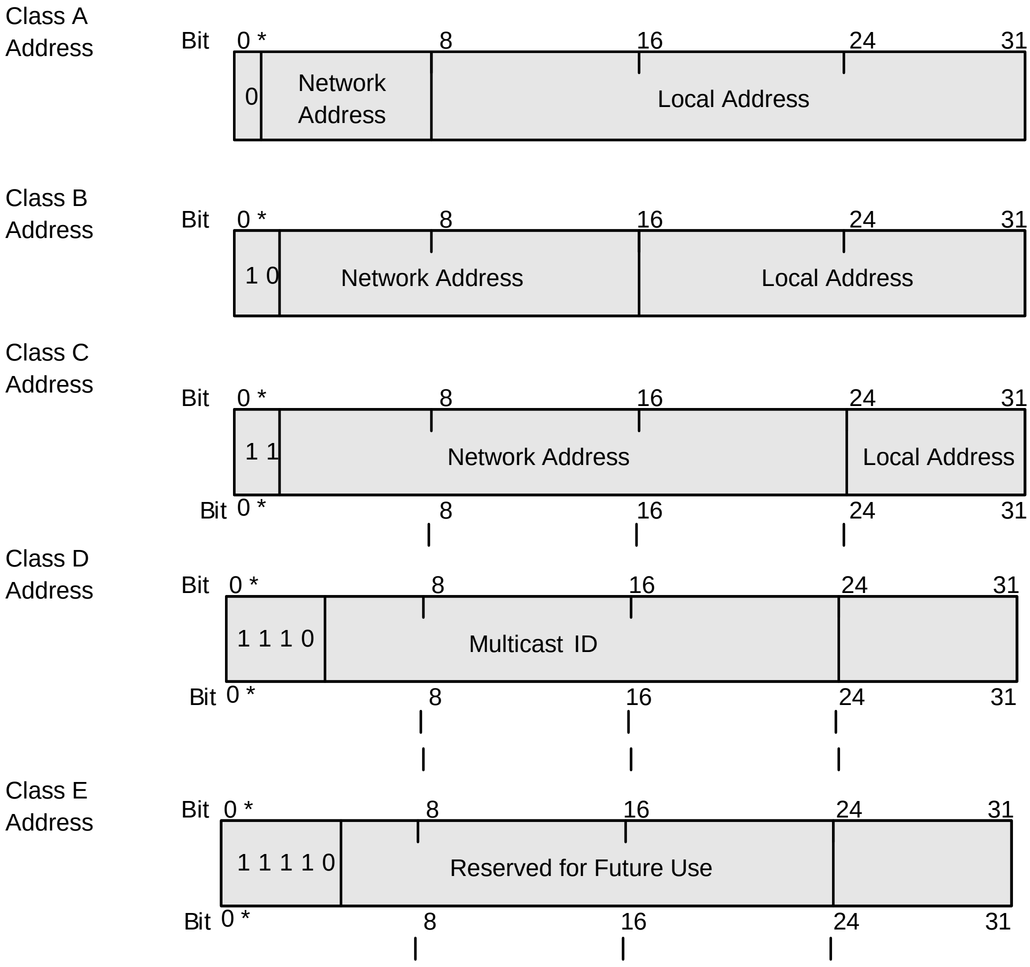
```
   DDN Network Information Center
   SRI International, EJ291
```

```
333 Ravenswood Avenue
Menlo Park, CA  94025
```

Assigned Internet addresses are Class A, B, or C depending on the value of the network address. If your LAN is standalone or private, your LAN administrator can choose a private set of Internet addresses. In that case, typical use calls for all Class A addresses.

As shown in Figure B-1, the class of the Internet address determines how the 32 bits are divided between the network address and the local address.

Figure B-1. Internet Address Format



Legend

* Note: Bit 0 is the most significant bit.

VST 015.VSD

A Class A address consists of a one-octet network address and a three-octet local address. The high-order bit of the network address is always 0; therefore, the first octet is a number in the range 0 through 127 (%H00 through %H7F). So, only 128 Class A networks can exist; however, each Class A network can have as many as 16,777,214 hosts, with addresses from %H000001 through %HFFFFFE (local addresses %H000000 and %HFFFFFF are reserved).

Note. The local address 0 is never assigned to an individual host. Internet addresses that have a local address of 0 are always refer to the network itself. The local address of all 1's is also never assigned to an individual host. Internet addresses that have a local address of all 1's are always reserved for broadcast.

A Class B address consists of a two-octet network address and a two-octet local address. The two high-order bits of the network address are always 10; therefore, the first octet is a number in the range 128 through 191 (%H80 through %HBF). So, 16,384 Class B networks can exist, each with as many as 65,534 hosts, with host addresses from %H0001 through %HFFFE (local addresses %H0000 and %HFFFF are reserved).

A Class C address consists of a three-octet network address and a one-octet local address. The three high-order bits of the network address are always 110; therefore, the first octet is a number in the range 192 through 223 (%HC0 through %HDF). So, 2,097,152 Class C networks can exist, each with 254 hosts, with host addresses from %H01 through %HFE (local addresses %H00 and %HFF are reserved).

A Class D address is a 4-octet multicast group address. The four high-order bits of the address are always 1110; therefore, the first octet is a number in the range 224 through 239 (%HE0 through %HEF). This means that an Internet can have a total of 268,435,456 multicast groups.

Class E addresses (240.0.0.0 to 247.255.255.255) are not currently supported and are treated as Class C addresses.

Broadcasting Using IP Addresses

As described above, a local address (hostid) consisting of all 1's is reserved for broadcast messages. The Internet protocols normally restrict the broadcast to the network on which the broadcast message originates. The relative efficiency of broadcast messages depends on the network technology employed. Some networks do not support broadcast messages, while others (such as Ethernet) handle them just like any other transmission. You should be aware of the configuration of your network and how it handles broadcast messages before you attempt to use them.

The Problem of Resolving Addresses

In the previous subsection, we discussed the concepts of Internet addressing. In practice, these concepts can be very complex to implement. One of the complexities of implementation involves the mapping of Internet (IP) addresses to hardware addresses. The mapping of IP addresses to hardware addresses and how that is achieved is the topic of this subsection.

Encapsulation of Messages

Because TCP/IP uses encapsulation, (placing a message of a higher layer into the data portion of the lower layer which then adds its own header), we cannot rely solely upon IP addresses to communicate with other machines on the network. When the network interface is assembling its own frame to place onto the medium, it adds its own header information which must contain, not an IP address of the destination, but a hardware destination address. The IP address of the destination is buried in the data of the network interface frame.

Static Binding of Addresses

Some networks use tables that include both the hardware and IP addresses. Other networks encode the hardware addresses into the IP addresses. In either case, this method of mapping IP addresses to physical addresses is known as static address binding. It is static because the information is static or when the information does change, it is done manually. Changes to this information are usually accomplished by updating a table that may be stored in an edit file.

The NonStop Implementation of TCP/IP over X.25 uses static address binding.

Dynamic Binding of Addresses

The NonStop implementation of TCP/IP over an Ethernet network or an IEEE 802.3 (SNAP) network avoids the limitation of static address binding by using the Address Resolution Protocol (ARP). The ARP is fully described in RFC #826 of the Internet standards.

Once a host receives information relating to the IP and hardware addresses of other hosts on the network, that host stores the information in cache. This is known as dynamic address binding. Since most Internet traffic consists of communication between pairs of machines, the cache eliminates the need for most ARP requests.

Consequently, this Internet-to-physical address information is stored in each host and/or router that is attached to the Internet. But doesn’t this result in a large number of addresses stored in all machines on the Internet? In fact, it doesn’t because ARP is used only on the local physical network. Thus, only those hosts and gateways/routers that are directly connected to the same physical network share one another’s Internet-to-physical address information.

Subnet Addresses

In the model, each network can be divided into subnetworks, or subnets. Within a given network, each subnet is treated as a separate network; outside that network, the subnets appear as part of a single network. Each subnet attached to a host is reached through a device (or a pseudo-device, such as a loopback driver) that provides an interface for IP data transmission. In this manual, the term subnet also refers to the interface through which the subnet is reached.

To address subnets within your LAN, your LAN administrator can further divide the local address part of your Internet addresses into a subnet number (identifying a particular subnet) and a host number (uniquely identifying a host system within the subnet).

To identify what part of the Internet address represents the subnet number, a 32-bit subnet mask is used. All bits corresponding to the network address and its subnet address are set to 1, and all bits corresponding to the remainder of the local address are set to 0.

Figure B-2 shows an example of a Class B address in which the first five bits of the local address are the subnet number, leaving the last eleven bits to identify the host. As shown, the subnet mask consists of twenty-one 1's (corresponding to sixteen bits of network address and five bits of subnet address) followed by eleven 0's.

Some of the remaining bits of the local address can define further subnets within each first-level subnet, as shown in Figure B-3. In this example, the 11-bit host number from Figure B-2 is subdivided into a 3-bit subnet number and an 8-bit host number. The subnet mask at this level consists of twenty-four 1s and eight 0s.

Figure B-2. Subnet Mask

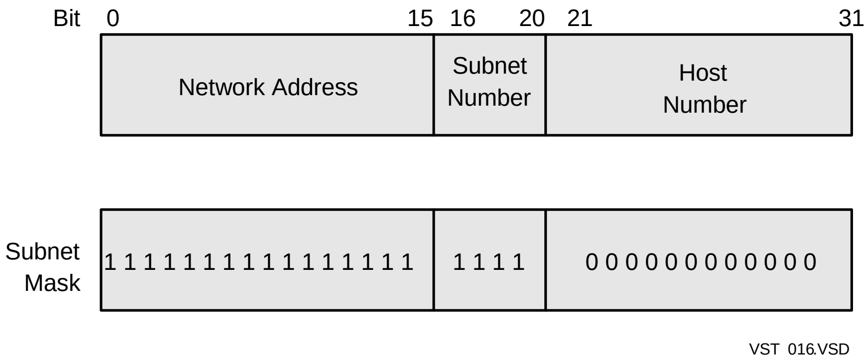
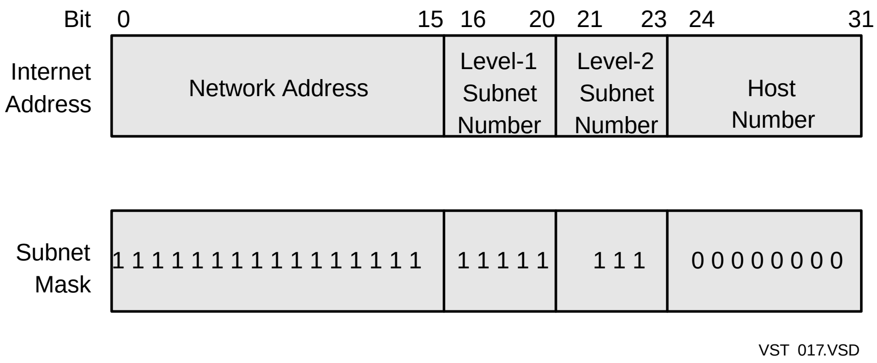


Figure B-3. Addressing of Nested Subnet



Subnet Addressing Advantages

There are many advantages of using subnet addressing. For one thing, it is transparent to the rest of the Internet. That is, the rest of the Internet does not know or care whether you have implemented subnet addressing on your own network. All routing and delivery mechanisms remain unchanged on the Internet.

Another advantage is that subnets may also be assigned and administered by the local administration authority. When you are provided with a network address, you may assign the subnet structure to suit your particular needs. As administrator of your local network, you design the subnet structure to support the networks and hosts on your network. This is discussed in more detail.

Another advantage of subnetting is that the existence of subnets does not increase or decrease the number of hosts that may be connected to any particular network. Consequently, if you are assigned a Class C network address, you are able to connect a total of 254 hosts to that network regardless of whether or not you have implemented subnet addressing.

In order to make this concept of subnet addressing more tangible, let’s look at an example of subnetting. As you know, there are three primary classes of Internet (IP) addresses: A, B, and C. As we have seen, the difference between these classes is that the network address and the local address are different sizes. Class A addresses allow one octet for the network address and three octets for the local address; Class B addresses allow two octets for the network address and two octets for the local address; and Class C addresses all three octets for the network address and one octet for the local address.

Subnetting can be used with all of the network address classes even though only Class B addresses are used in the examples that follow. Figure 6-1 illustrates the use of a Class B address. Notice how subnetting allows you to split up the two octets of the local address into an actual physical network address (128) and a local address (3).

Using the previous example, the physical network could be arranged like the network shown inFigure B-2.

This illustration shows how the two octets of the local address can be split so that the first octet specifies the physical network address (128, 64, and 192 respectively) while the second octet is used to specify the physical host address.

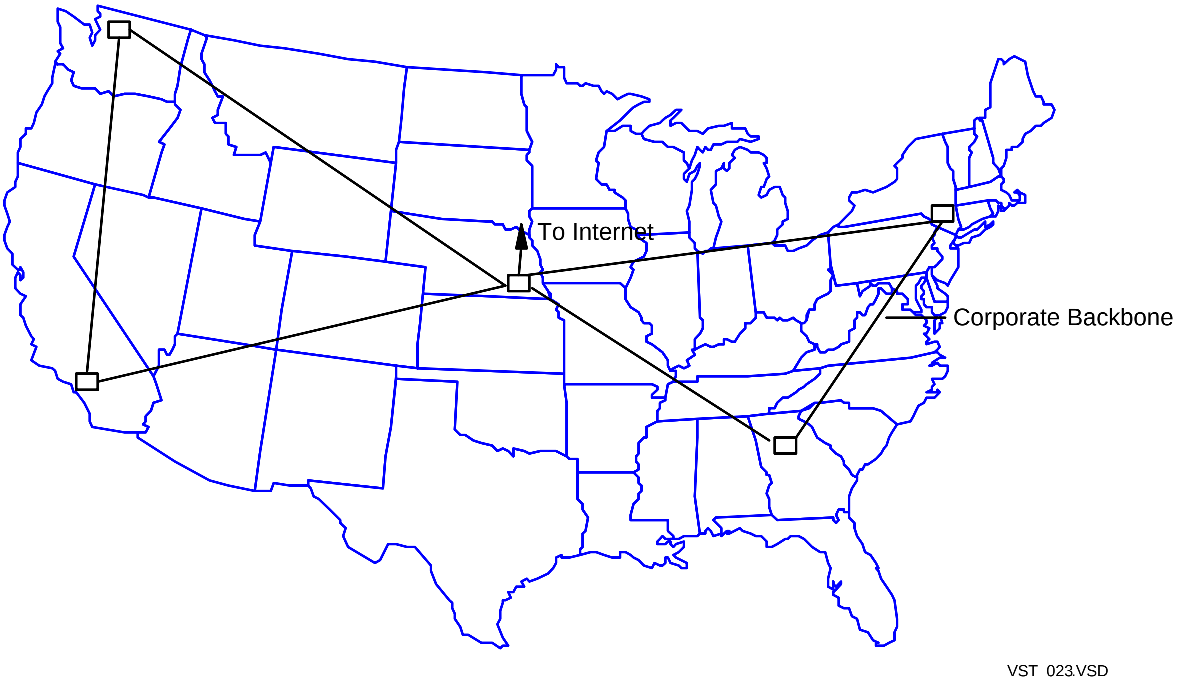
These different physical networks could be in the next room, an adjoining building or across the country. It therefore makes sense to subnet a network when there are geographic or administrative reasons to do so.

You can see from this example that from a single Class B network address, the administrator can separate the last two octets of the local address into a physical network address and a local host address. The same concept applies to any class of Internet addresses you may have.

Example: Configuring an Intranet Using Subnetting

Your company is headquartered in Omaha and has branch offices in Atlanta, Los Angeles, Seattle, and New York. The headquarters has 100 hosts, Atlanta has 17, Los Angeles has 30 hosts, Seattle has 12 hosts and New York has 65 hosts. The objective is to design the subnet masking scheme to accommodate all the offices and to allow for growth. Figure B-4 shows the Intranet described in this example.

Figure B-4. Example of a Corporate Intranet



Task 1: Apply for a Class B Internet Network Address

Their address is:

```
DDN Network Information Center
SRI International, EJ291
333 Ravenswood Avenue
Menlo Park, CA  94025
```

In this example, assume that the class B Internet address assigned to Omaha is 130.10.128.3

Task 2: Create Subnets for Each Office Network

1. In this case you have more hosts than physical networks; divide the two octets of the local address such that more bits are available for hosts. Table B-1 shows that the maximum bits you need for hosts is 10; therefore, you need 10 bits to represent hosts. This leaves the remaining six bits in the local address portion (as shown in Figure B-5) to indicate a subnet. Six bits allows for up to 64 subnets, each with a maximum of 1,024 hosts.

Table B-1. Subnet Addressing Based on Current and Projected Needs

	# Networks	#Hosts	Required Room for Growth	Required Bits for Networks	Required Bits for Hosts
Omaha	1	100	1,000 hosts, 25 networks	6	10
Atlanta	1	17	100 hosts, 2 networks	1	7
Los Angeles	1	30	100 hosts, 2 networks	1	7
Seattle	1	12	50 hosts, 2 networks	1	6
New York	1	30	100 hosts, 2 networks	1	7

2. Divide the local addresses to accommodate the needs identified in Table B-1. Use six bits of the first local address octet for network (subnets) addresses, two bits of the first octet and all eight bits of the second octet for the host addresses. Figure B-5 shows the class B IP address, with the local address occupying two octets.

Figure B-5. A Class B IP Address

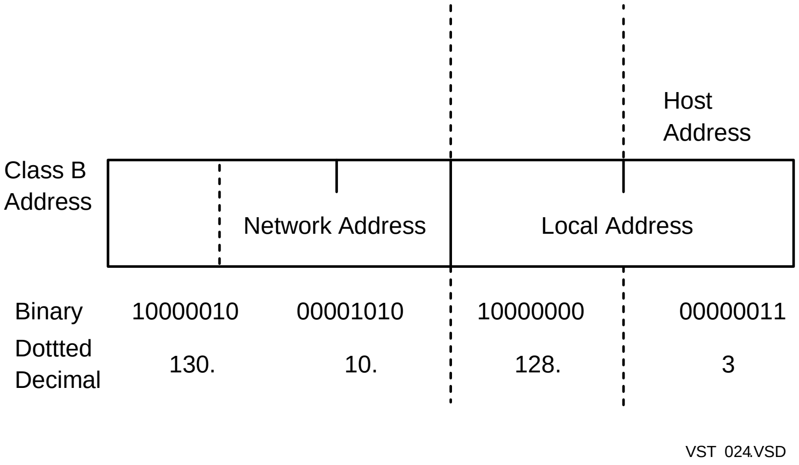
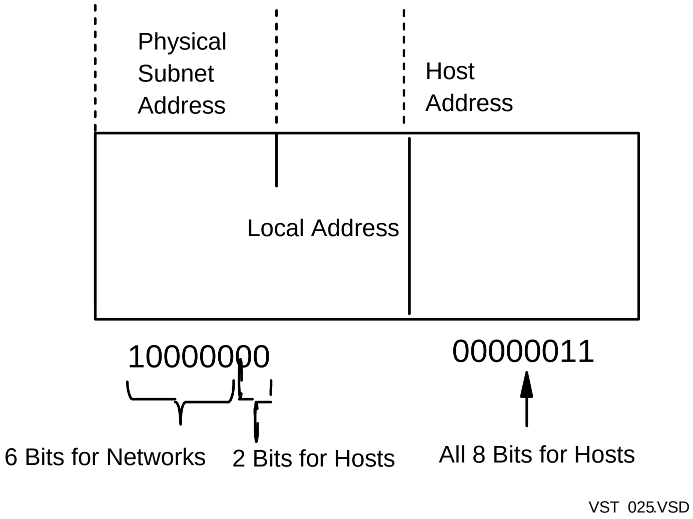


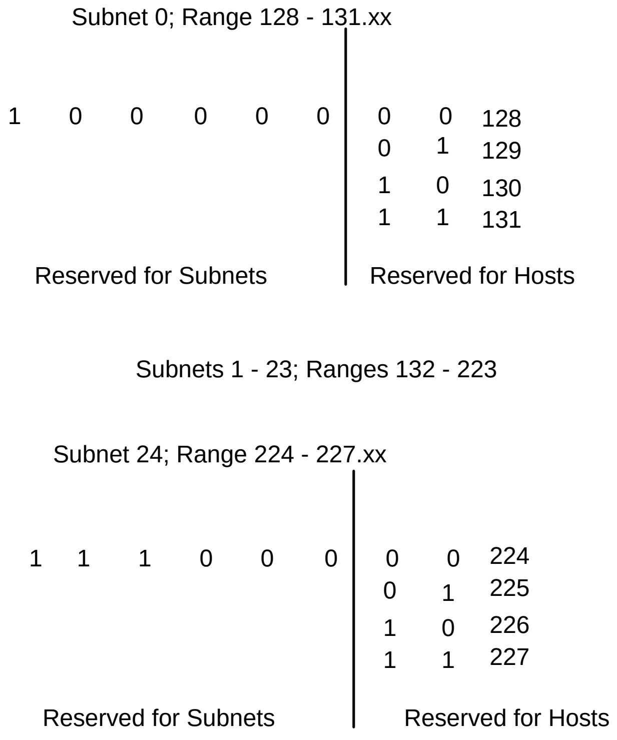
Figure B-6 shows the local address divided into six and eight bits.

Figure B-6. Splitting a Class B IP Address for Subnetting



1. Allocate the subnets to each of the offices. Omaha has subnets 0-24, Atlanta has 25-26, Los Angeles has 27-28, Seattle has 29-30, and New York has 31-32. Figure B-7 shows Omaha’s subnets. Since Omaha’s range ends at 227, Atlanta’s range is 228-235, Los Angeles’s range is 236 - 241, Seattle’s range is 242-247, New York’s range is 248-253.

Figure B-7. Omaha’s Subnets



VST 026.VSD

2. Once you’ve allocated the subnets, each office can assign their own local addresses from within the range allocated to them for the subnet.

3. Once you’ve split the class B IP address for subnetting, determine the subnet mask. Each ADD SUBNET command requires a subnet mask. The routing algorithm residing in each host implements an OR operation on the subnet mask and the destination to eliminate all but the relevant bits in the address. In all cases, the first two octets are relevant; in our sample case, the first six bits of the second octet are also relevant because they represent subnets. The mask is therefore: 11111111.11111111.11111100.00000000 (255.255.252.0). Figure B-8 shows the IP address that results from the mask applied to the first subnet (Omaha’s). The resulting IP address is 130.10.128.

Figure B-8. The Or Operation on the Mask and IP Address

```
      10000010.00001010.10000000.00000011
or    11111111.11111111.11111100.00000000
---------------------------------------------
      10000010.00001010.10000000.00000000
        130       10        128
```

VST 027.VSD

4. Translate the subnet mask to hexadecimal format. Break each byte into two nibbles (four bits) and assign its hexadecimal value (hexadecimal ranges from 0 to F):

```
1111 1111 1111 1111 1111 1100 0000 0000

F    F    F    F    F    C    0    0
```

5. Add the subnets for Omaha. For examples of command files, see Section 1, Configuration Quick Start.

Networking Technologies

The term interoperability describes the cooperation of different machines in solving a common problem. Interoperability requires that these machines communicate with each other, share or exchange data, and execute a variety of computational tasks. To achieve interoperability, the protocol set has to operate with many different networking technologies.

NonStop TCP/IP runs on the NonStop operating system. NonStop TCP/IP allows heterogeneous systems in a multinetwork environment to communicate with each other.

NonStop TCP/IP software conforms to a number of Request for Comments (RFCs), Internet Engineering Notes (IENs), and Military Standards (MIL-STDs) maintained by the Defense Data Network (DDN) at the DDN Network Information Center (NIC) operated by SRI International in Menlo Park, California. The RFCs, IENs, and MIL-STDs define the protocols implemented by the NonStop TCP/IP software.

Conformance to DDN specifications includes operation of the Internet Protocol (IP) over Ethernet LANs.

Managing NonStop TCP/IP Configuration Files

Two types of configuration files are associated with the NonStop TCP/IP subsystem:

- A configuration command file (startup file)
- Configuration files for NonStop TCP/IP protocols and services

For information on startup command files, see Section 1, Configuration Quick Start and Section 3, Configuring the NonStop TCP/IP Subsystem. For information on the configuration files for protocols and services, see Section 3, Configuring the NonStop TCP/IP Subsystem.

Retrieving Configuration Information

SCF provides several commands that retrieve configuration information for the objects associated with the NonStop TCP/IP subsystem. The INFO, LISTOPENS, NAMES, STATS, STATUS, and VERSION commands all return useful configuration and/or operation data, as follows:

- The NAMES command displays all the names currently configured for a specific object type (SUBNET, ROUTE, or WINDOW).
- The INFO SUBNET command displays the subnet name, the name of the SLSA line, the subnet's Internet address, the subnet type, and the subnet mask.
- The INFO ROUTE command displays the route name, the name of the associated subnet, the Internet addresses of the destination and the gateway, and the route type.
- The Hint on page 1-7 explains how to use SCF to access information about the LIFs in the SLSA subsystem.

These commands and responses are explained in detail in Section 4, SCF Reference.

Listing all Running NonStop TCP/IP Processes

Since there is no NAMES PROCESS command, to obtain a list of all running NonStop TCP/IP processes, enter one of the following commands at the SCF prompt:

```
-> LISTDEV TYPE 48
```

or

```
-> LISTDEV TCPIP
```

This command provide the following response display:

```
LDev Name      PPID    BPID   Type     RSize Pri Program
578  $ZTC0     8,301   9,322  (48,0 ) 32000 200 \SNAX.$SYSTEM.SYS02.TCPIP
585  $ZTC1     5,329   6,313  (48,0 ) 32000 200 \SNAX.$SYSTEM.SYS02.TCPIP
589  $ZTC2     1,310   2,321  (48,0 ) 32000 200 \SNAX.$SYSTEM.SYS02.TCPIP
593  $ZTC3     4,329   7,313  (48,0 ) 32000 200 \SNAX.$SYSTEM.SYS02.TCPIP
```

Socket Programmatic Interface

The socket interface, provided as part of the NonStop TCP/IP software, uses Guardian file-system procedures for interprocess communication and provides socket library routines. Application processes request the NonStop TCP/IP software to create a socket when needed; the application specifies the type of service desired. Applications can request TCP, UDP, and raw sockets for use with IP. The NonStop TCP/IP software returns a socket number that the application uses to refer to the new socket. The application then (optionally) binds the socket to a specific local address and port, and sends or receives data on the socket. When the transfer of data is complete, the application can (optionally) shut down the socket and destroy (close) it.

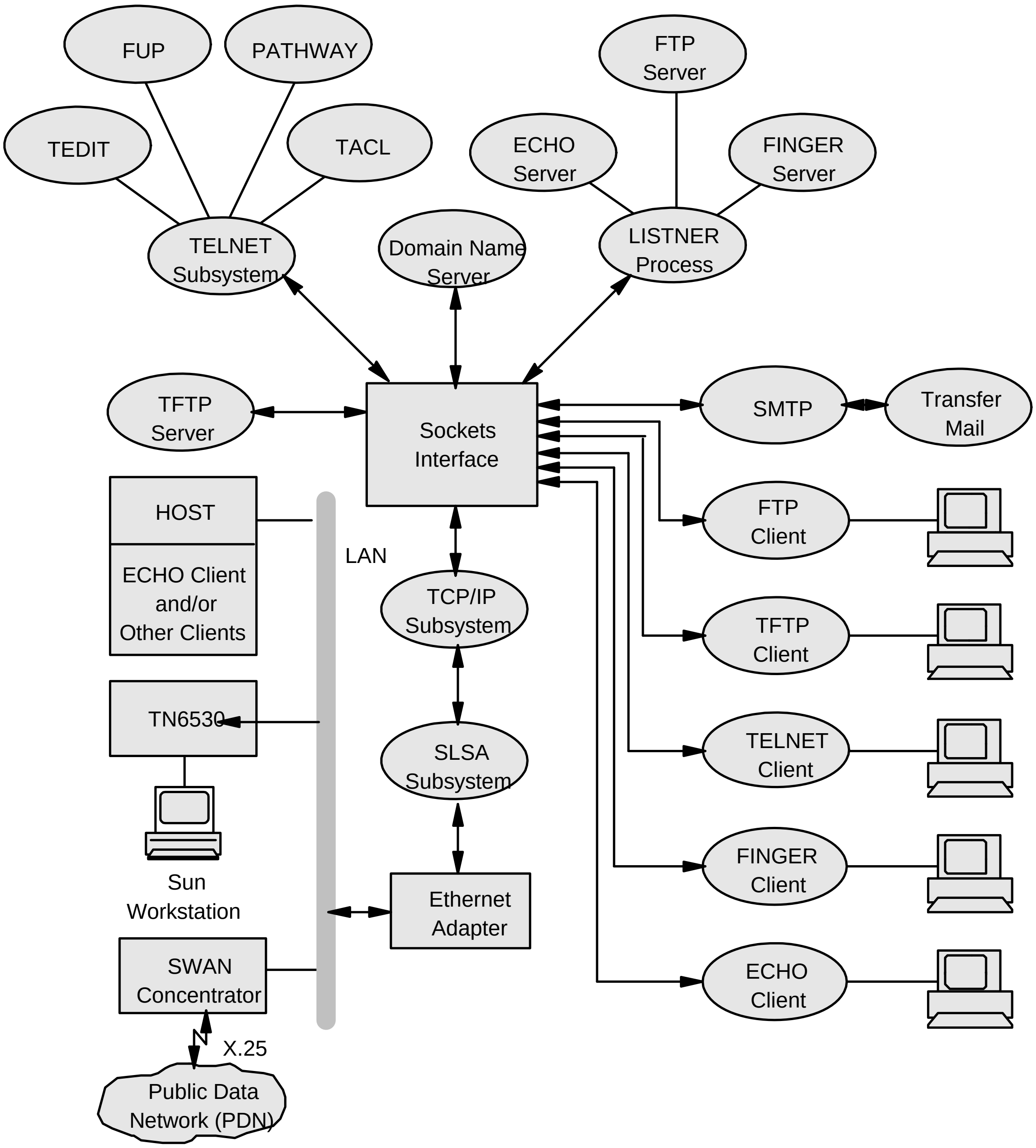
The socket library routines are based on the socket programmatic interface primitives in the 4.3BSD release of the UNIX operating system. However, the socket library routines include extensions to adapt the Berkeley socket interface to Guardian operating-system features such as nowait I/O.

The socket interface is modeled after the BSD socket interface to facilitate porting of existing UNIX TCP/IP applications to a NonStop system. For a detailed description of the socket library routines and a summary of the differences between the NonStop TCP/IP socket interface and the 4.3BSD UNIX interface, refer to the *TCP/IP Programming Manual.*

NonStop TCP/IP Processes and Protocols

The NonStop TCP/IP subsystem has a wide range of capabilities provided by a number of individual components, acting alone or together. Figure B-9 shows a detailed view of the NonStop TCP/IP subsystem's internal processes and components. The next subsection describes the components shown in Figure B-9.

Figure B-9. NonStop TCP/IP Subsystem Processes



Supported Protocols

NonStop TCP/IP supports the following processes and protocols:

- NonStop TCP/IP Subsystem Processes
  - NonStop TCP/IP Process
  - TELSERV Process

    - LISTNER Process
- NonStop TCP/IP Protocols
    - Transmission Control Protocol (TCP)
    - User Datagram Protocol (UDP)
    - Internet Protocol (IP)
    - Internet Control Message Protocol (ICMP)
    - Address Resolution Protocol (ARP)
    - Subnetwork Access Protocol (SNAP)
    - File Transfer Protocol (FTP)
    - Trivial File Transfer Protocol (TFTP)
    - Simple Mail Transfer Protocol (SMTP)
    - ECHO Server/Client
    - FINGER Server/Client
- Other Services
    - TN6530 Emulation Utility
    - Socket Programmatic Interface (BSD 4.3)
    - Domain Name Server (DNS)
    - Domain Name Resolver

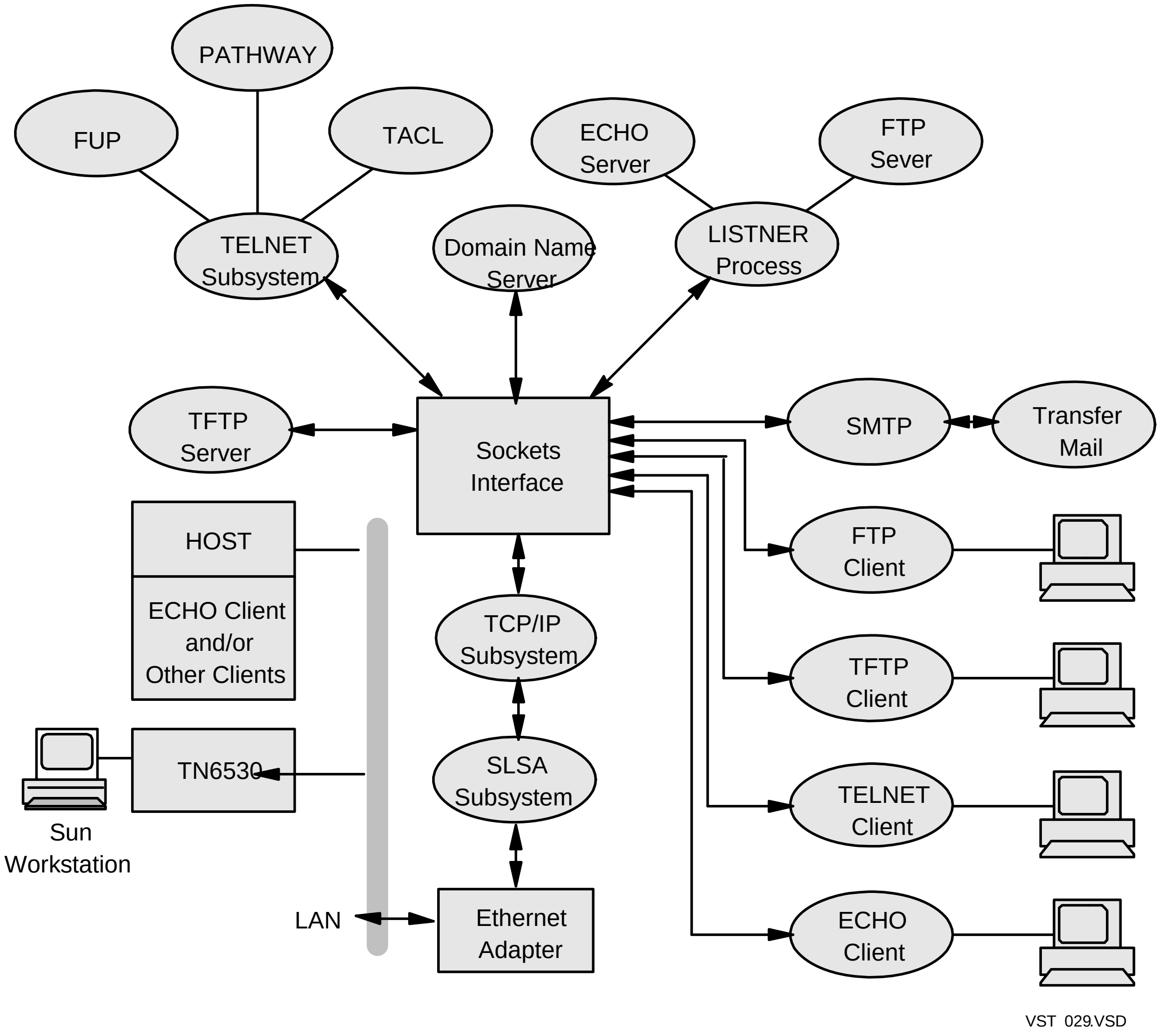
The terms client and server are used here as they are customarily used in TCP/IP literature. A server accepts requests, performs specified services, and returns results to the requesters over the network. A client sends requests to the server and waits for the server to respond. The client-server model is the same model known as the requester-server model in other documentation for NonStop systems. A client is the same as a requester.

NonStop TCP/IP Subsystem Processes

The NonStop TCP/IP subsystem processes are described in detail in the paragraphs that follow. NonStop TCP/IP software conforms to the RFCs, IENs, and MIL-STDs that define those protocols implemented. The descriptions that follow specifically point out any nonconformance to the protocols and standards required by the NonStop operating system.

The NonStop TCP/IP subsystem is composed of both servers and clients (Figure B-10). A client is a process that sends requests to the server and waits for a response. A server is a process that offers a service that can be used over the network; it accepts requests, performs the specified services, and returns the results to the clients. All supported servers and clients are described later in this section.

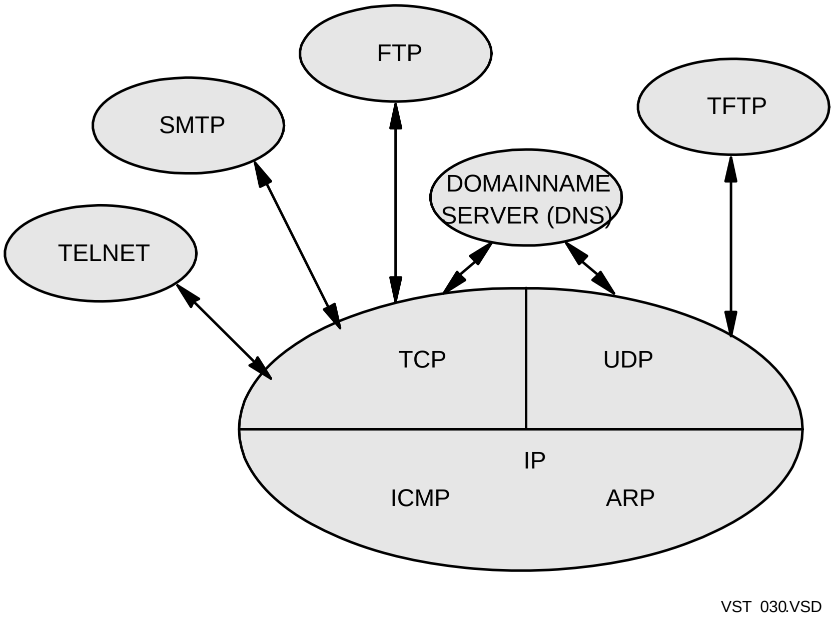
Figure B-10. NonStop TCP/IP Processes and Protocols



NonStop TCP/IP Process

The NonStop TCP/IP process (Figure B-11) is a single process that performs basic services for FTP, TFTP, SMTP, ECHO, FINGER, and TELNET client and server programs. Multiple NonStop TCP/IP processes can run on a system; however, in most cases, only one NonStop TCP/IP process runs on a system even if the system has multiple networks attached. Each NonStop TCP/IP process is a Guardian named process. NonStop TCP/IP process names have the format $ZTC*n*, where *n* is an integer (usually 0 through 9).

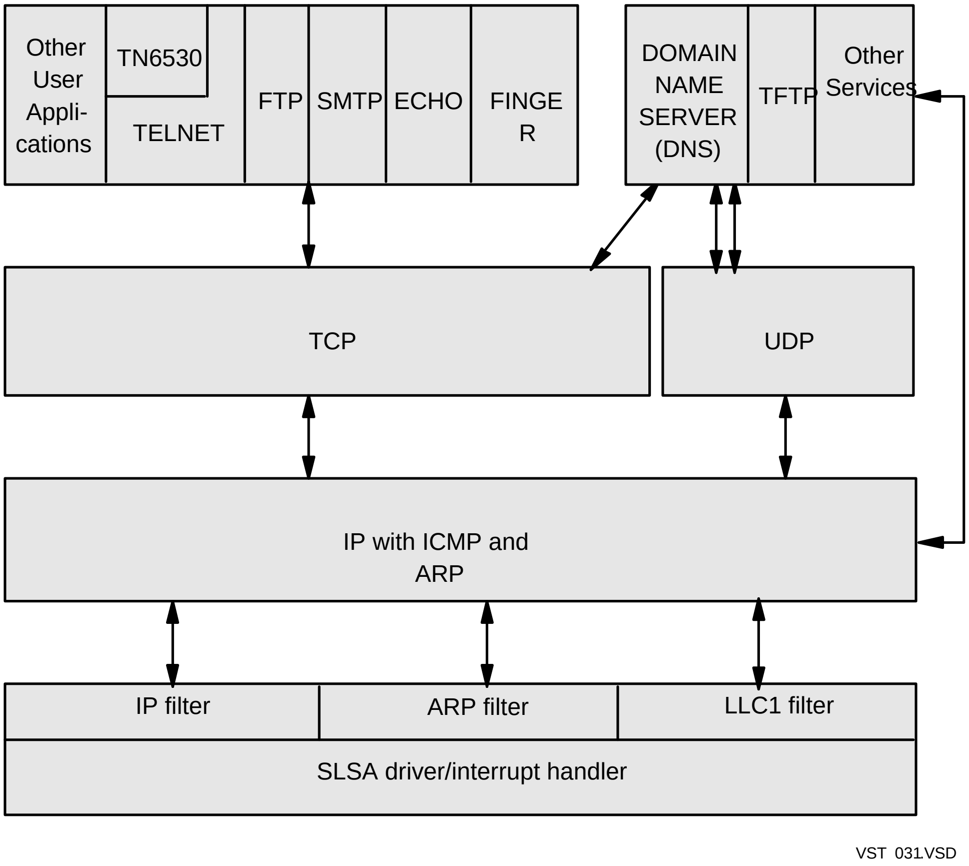
Figure B-11. NonStop TCP/IP Process



NonStop TCP/IP Layered Architecture

The NonStop TCP/IP product provides a layered architecture that includes many commonly used TCP/IP applications (as shown in Figure B-12). Domain Name Server uses UDP for queries and TCP for messages such as zone transfers.

Figure B-12. Layered Architecture of NonStop TCP/IP Software



NonStop TCP/IP as a NonStop Process Pair

Run the NonStop TCP/IP process as a NonStop process pair. When running the process as a NonStop pair, the primary process attempts to start the backup process as it completes its initialization. Should the attempt fail, the primary process delays for a period, then attempts to start the backup process again. After each attempt, the primary process delays for a slightly longer interval prior to attempting the restart until the interval reaches a maximum value of 10 minutes.

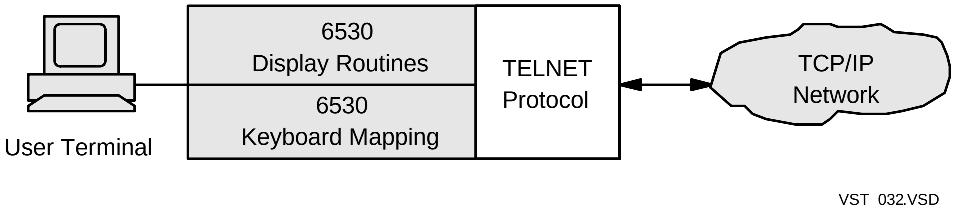
The primary NonStop TCP/IP process keeps the backup process ready to take over by checkpointing all configuration changes and socket creation, and deletion and state changes. A NonStop TCP/IP process provides a persistent TCP/IP process, which recovers from processor failure and SCF primary requests.

If the backup process of a TCP/IP NonStop pair abends for any reason, the primary process tries to restart the backup process after a delay. This delay increases after each failure until it reaches a maximum of 10 minutes. An EMS message is issued when the backup process has abended and displays the time after which the primary process attempts to create the backup. The period for recovery initially begins at 5 seconds with a maximum delay of 10 minutes. The period is reset to the initial value each time the backup processor is reloaded. When the backup process abends, it creates a save abend dump file. The save file created only contains the stack area of the process and not the full QIO segment. If the backup abends during its initialization code, no save abend file is created.

TN6530 Emulation Utility

The multiple-page terminal emulation utility (TN6530) is a UNIX TELNET client that interfaces to NonStop TCP/IP and TELSERV. TN6530 emulates a 6530 terminal, including its block-mode support. TN6530 allows a user on a UNIX system to connect to a NonStop system and run applications that require a multiple-page terminal such as the 6530. Figure B-13 shows the software layout of the TN6530 utility and its relationship to the Internet.

Figure B-13. TN6530 Software Layout and Relationship to Internet



Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) is used by applications that require reliable end-to-end data transfer. It is a stream-oriented protocol that has no concept of packet boundaries. TCP guarantees that all data sent will be received and will arrive in the same order in which it was sent.

The NonStop TCP/IP process provides a socket interface to the Transmission Control Protocol (TCP, RFC 793), which uses Guardian file-system procedures for interprocess communication. TCP is a stream-oriented protocol with no concept of record boundaries.

Application processes call a socket routine to request that the NonStop TCP/IP software create a socket when needed; the application specifies the type of service.

Port Numbers

Both TCP and UDP use a 16-bit port number to select a socket on the host. Client programs normally use more-or-less random port numbers; however, specific port numbers (called well-known ports) are assigned for use by server programs. Each well-known port is associated with a specific service. A client requesting a particular service specifies as the destination port the well-known port associated with that service. The server program listens at that port for requests. The well-known port numbers are listed in Appendix C, Well-Known Port Numbers for TCP and UDP.

TCP and UDP allow IP to run several simultaneous sessions with a given host. Multiple sessions are accommodated by specifying a port number, which identifies the communication path, along with the Internet address. Each end of the communications path is assigned a port number for that session.

Applications using TCP and UDP perform addressing by specifying a unique combination of a destination Internet address, destination port number, source Internet address, and source port number. In TCP, this combination uniquely identifies a connection and is part of every TCP packet that goes over the Internet.

Generally, at least one end of the conversation asks for a port number that is guaranteed to be unique. The client program normally requests a port number, because the server typically uses a well-known port.

User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) provides unreliable datagram service. The integrity of the packets sent is maintained; when a packet is received, it is guaranteed to exactly match what was sent. However, the delivery of packets is not guaranteed, and there is no guarantee of the order in which datagrams are received.

A datagram is designed for connectionless protocols such as IP. A connection is not established before the message is sent; thus, each datagram includes the address of the destination host, as well as the source.

As for TCP, (described in Transmission Control Protocol (TCP) on page B-20), application processes call a socket routine to request the NonStop TCP/IP software to create a UDP socket when needed; the application specifies the type of service desired. The TCP and UDP protocols assume that the Internet Protocol (IP) is used for network layer services. Like IP, UDP is a connectionless protocol; it uses datagrams. TCP, on the other hand, is a connection-oriented protocol, requiring a connection to be established before messages are sent.

TCP messages specify the connection with which they are associated. For UDP, the same combination of addresses and numbers described for TCP identifies a temporary source and destination, rather than a connection. This combination is a part of every UDP packet that goes over the Internet.

For a detailed description of the socket library routines available, refer to the *TCP/IP Programming Manual*. For more detailed technical information on the UDP, refer to RFC 768 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-175 through 2-178). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

Internet Protocol (IP)

The Internet Protocol (IP) accommodates the connectionless delivery of Internet datagrams between heterogeneous networks. The IP also services various host-to-host protocols. The IP provides many capabilities at the network level, and forms the foundation of the NonStop TCP/IP product. The TCP and UDP protocols use the IP protocol. Applications can build their own transport-layer protocols directly on this protocol by using raw sockets.

Message Routing

In the IP, messages called datagrams pass from a source (host) to a destination. An IP datagram may consist of one or several network-layer messages, or packets. Datagrams in the Internet often pass through a dozen different networks before reaching their final destination. The message routing is performed by the IP and is transparent to the application. The application only needs to specify its own local Internet address and the remote Internet address of the destination.

The IP uses routes, which are Internet paths, to transmit datagrams. A route specifies both a final destination and the gateway or host to which a datagram must be sent next to reach that destination. The IP layer at each host maintains a table of routes. The table lists the Internet address of each accessible network, or of a particular accessible host on a network, and the gateway used to get to that network or host.

To keep their routing tables simple, hosts often specify a default gateway or router to which datagrams can be sent if the routing table contains no entry for a particular network. The hosts thus rely on the gateways, which contain comprehensive routing tables, to determine the best path for the datagram.

Raw Sockets

As for TCP and UDP, described above, application processes call a socket routine to request the NonStop TCP/IP software to create a raw socket when needed. A raw socket provides direct access to the IP. Raw sockets allow applications to take advantage of protocol features not directly accessible through the TCP or UDP interfaces. Applications also can develop new protocols by using raw sockets.

Programming at the IP level by using raw sockets requires the application to provide support for the transport protocol used above IP. If your application program refers to a transport protocol by name, the program must provide an entry, including the protocol number and name, in the file $SYSTEM.ZTCPIP.PROTOCOLS, as described in Section 1, Configuration Quick Start.

For a detailed description of the socket library routines available, refer to the *TCP/IP Programming Manual.* For more detailed technical information on the IP, refer to RFC 791 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-99 through 2-150). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

Internet Control Message Protocol (ICMP)

The NonStop TCP/IP software uses the Internet Control Message Protocol (ICMP) with the IP to report errors and other control information from destination systems or gateways.

ICMP uses the Address Resolution Protocol (ARP) to convert 32-bit internet (IP) addresses into 48-bit Ethernet addresses. NonStop TCP/IP software uses the Internet Control Message Protocol (ICMP, RFC 792) along with IP to report errors and other control information from destination systems or gateways. The ICMP is considered a required part of any IP implementation.

For more detailed technical information on the ICMP, refer to RFC 792 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-151 through 2-172). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

ICMP Router Discovery Protocol

RFC 1256 defines the ICMP Router Discovery Protocol (IRDP) which uses ICMP messages on the host side to solicit default routes from routers on the local attached LAN and on the router side to advertise the availability of default routes. If multiple redundant routers are present, IRDP can detect secondary routes when a primary route fails. This secondary use of RFC 1256 is described by HP as Alternate Route Selection.

A default route is a path from a host to a router along which the host forwards IP packets that are destined for some non-local LAN. The router must figure out the path to the destination LAN. This method effectively delegates all routing responsibilities to the router, which is the recommended procedure for NonStop TCP/IP in most LAN configurations. Without IRDP, the NonStop TCP/IP administrator must configure manually a static default route to the specific IP address of the router. If the router's IP address changes, the configuration files on the NonStop machine must change also. With IRDP, there is no need to change any configuration files other than initially enabling the IRDP feature.

When route hold-down times and advertisement intervals are configured to approximately 30 seconds on the router, and there is no backup router on the LAN, IRDP can be used for dead-gateway detection.

When IRDP is enabled on a subnet, NonStop TCP/IP processes IRDP messages from multiple routers on the LAN and switches the default route to the most-preferred router that is currently advertising a default route. When one router stops advertising, another can take over. Typically, an application times out while the routers reconfigure their own routing tables to account for the lost router. If the application can tolerate a short time out, the TCP connection is maintained. For example, most Telnet implementations can tolerate time outs up to two minutes.

The NonStop TCP/IP process directly interprets incoming IRDP advertisements and updates its routing table by adding a default route to the advertising router. Adhering to RFC 1256, NonStop TCP/IP solicits default routes when new subnets are started. NonStop TCP/IP also solicits default routes when a multi-hop TCP connection begins to time-out. A time-out may be due to a router failure.

For more detailed technical information on the ICMP, refer to RFC 1256 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-151 through 2-172). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

Address Resolution Protocol (ARP)

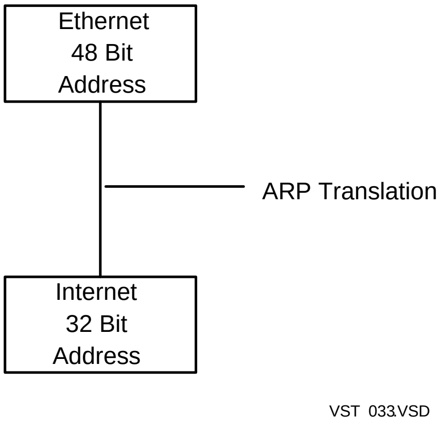
The Address Resolution Protocol (ARP) dynamically binds a high-level Internet Address to a low-level physical hardware address. Two machines on a single physical network must know each other’s physical address to communicate. The ARP’s address-resolution capability extends across only one physical network and is limited to networks that support hardware broadcast.

Ethernet (as originally defined by DEC, Intel, and Xerox (DIX)) allows many protocols to coexist on the same cable. Each protocol uses Ethernet packets (frames) that have their own type specified in the type field in the Ethernet packet header. The Ethernet specification requires the 48-bit physical (hardware) addresses that are assigned in blocks to hardware manufacturers by the Xerox Corporation.

The Internet Protocol (IP) specification requires the 32-bit Internet (IP) addresses that are assigned by the DDN Network Information Center (NIC) operated by SRI International in Menlo Park, California.

To use the IP over an Ethernet network (or any other network that uses a different size address), the two types of addresses must be resolved. The NonStop TCP/IP software uses the Address Resolution Protocol (ARP, RFC 826) to convert 32-bit Internet (IP) addresses into 48-bit Ethernet addresses. The ARP is a low-level protocol that effectively hides physical addresses and allows the IP to employ 32-bit Internet addresses. Figure B-14 shows ARP’s relationship to the Ethernet and Internet protocols.

Figure B-14. Address Resolution Protocol



The caller (host A) broadcasts a special packet addressed to the receiver’s Internet address (host B). All nodes receive the packet; however, only host B replies, and its reply includes its physical address. When host A receives this reply, it dynamically binds the Internet-to-physical address and caches it for subsequent use.

For more detailed technical information on the ARP, refer to RFC 826 (*DDN Protocol Handbook*, *Volume 3*, DDN Network Information Center, December, 1985, pp. 3-615 through 3-624). Also, refer to the book *TCP/IP Illustrated* by W. Richard Stevens, Prentice Hall, 1994.

Subnetwork Access Protocol (SNAP)

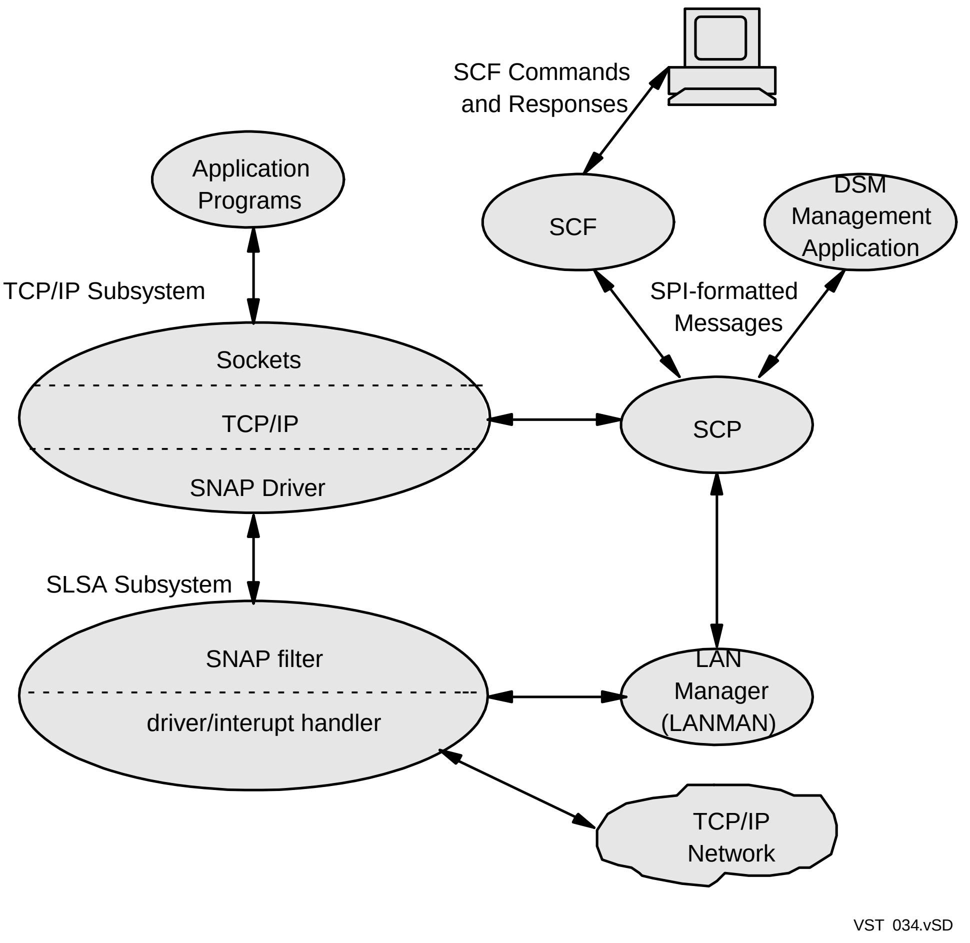
The Subnetwork Access Protocol (SNAP) defines an interface between the IP layer and the Logical Link Control (LLC) layer. This interface is required to run the TCP/IP protocol suite over IEEE 802.3 (Ethernet) LANs.

The IEEE 802.2 LLC type 1 is used for accessing the LAN. The interface is accomplished by an extension of the LLC header that contains a predefined Service Access Point (SAP) for use in the Source SAP (SSAP) and Destination SAP (DSAP) fields of the LLC header.

The IEEE has defined one data-link layer standard (IEEE 802.2) and several physical-layer standards (IEEE 802.3, 802.4, and 802.5) for LANs. The physical-layer standards specify the physical and Media Access Control (MAC) sublayer requirements for several different LANs, including IEEE 802.3 CSMA/CD LANs.

As shown in Figure B-15, the SNAP is part of the TCP/IP process and has an interface driver built into it.

Figure B-15. Subnetwork Access Protocol (SNAP) Interface



The SNAP driver also is responsible for distributing incoming packets to the IP and ARP software. Incoming packets are delivered to the SNAP driver from SLSA in the form of AGSDUs. The SNAP driver separates the individual messages and strips off unnecessary protocol headers before delivering messages to the IP and ARP input routines.

For more detailed technical information on the SNAP, refer to RFC 1042 (available on the Internet at: http://ds.internic.net/ds/rfc-index.html.)

Trivial File Transfer Protocol (TFTP)

The Trivial File Transfer Protocol (TFTP, IEN 113) client and server programs run over a UDP socket. The TFTP programs are not connection-oriented, and can be used only to retrieve or store publicly readable and writable files. The TFTP server listens for requests on the TFTP port (UDP port 69). More than one file can be transferred simultaneously.

The TFTP is divided into client and server programs, like other Internet applications. These two programs enable you to retrieve a file from or store a file on a remote system. Because these programs are not connection-oriented, and they do not use any user-authentication mechanism, you must place some restrictions on the files that can be sent and received from a remote system.

The TFTP server imposes certain restrictions when you retrieve a file from a NonStop system. You cannot retrieve a file unless it is readable to the network (unless its access security is NXXX). The write/execute/purge permissions may be anything.

The TFTP server also imposes certain restrictions when you try to store a file on a NonStop system. The file must already exist on the NonStop system, and it must be network writable (its access security is XNXX).

The system manager can place further restrictions on the machine in which the TFTP server is running by specifying public volume.subvolume names on the command line when starting the TFTP server. If the names are specified, files can be retrieved from and stored on the specified volume.subvolume. Up to six separate subvolumes can be specified.

New files may be created in the specified subvolumes. However, if a file already exists, it must be secured XNXX if it is to be written to. New files are created by the TFTP server with the permissions NNNG. These permissions allow remote users to overwrite the new files.

The TFTP client provides the user interface to the Internet TFTP. A TFTP client may store files virtually anywhere, depending only on the client’s login restrictions.

For a detailed description of the operation of the TFTP server and client, refer to the *TCP/IP Applications and Utilities User Guide*. For more detailed technical information on the TFTP, refer to IEN 133 (*DDN Protocol Handbook*, *Volume 2*, DDN Network Information Center, December, 1985, pp. 2-965 through 2-984). Also, refer to the book *Internetworking with TCP/IP* (Douglas E. Comer, Prentice Hall, 1991).

Simple Mail Transfer Protocol (SMTP)

The Simple Mail Transfer Protocol (SMTP) is a protocol implemented on a NonStop system that provides an interface between TRANSFER Mail systems and SMTP hosts. The SMTP protocol enables TRANSFER Mail users (referred to as TRANSFER hosts) to send and receive messages to users on SMTP hosts.

The SMTP protocol also relays SMTP messages between SMTP hosts by acting as an intermediate node that stores and forwards SMTP messages.

TRANSFER

The SMTP gateway does not deliver mail directly to your terminal; it uses TRANSFER as the vehicle for local delivery and also for storing mail that has to be forwarded. TRANSFER is a high-level software product that reliably supports communication between users, devices, and processes. TRANSFER maintains a database that describes users and provides holding areas for packages.

The SMTP gateway also can relay SMTP messages between SMTP hosts. Acting as an intermediate node, it can store and forward SMTP messages. Messages received by the gateway from one SMTP host to be relayed to another SMTP host are first sent to TRANSFER.

TRANSFER treats the SMTP gateway as a mail correspondent (SMTPGATE). Messages from TRANSFER hosts to SMTP hosts, or from one SMTP host to another SMTP host, are sent to the gateway's mail correspondent ID on the gateway node. the address suffix contains the SMTP host destination path. TRANSFER messages are addressed as follows:

```
correspondén_name [@node] [(suffix)]
```

The SMTP processes SMTPRCV and SMTPSND (described below) first open the T1SERV (Transfer Process) $ZCC0. They then start a session with TRANSFER by logging in as the SMTP MAIL gateway correspondent. Once the processes log into TRANSFER, they can make requests to TRANSFER to store, retrieve, or delete messages. The mail correspondent name and password are specified in the SMTPCONF configuration file, as described in Configuration Files in the Guardian Environment on page 3-34.

Note. Because the password is specified in the configuration file, be sure to secure it so users cannot access the SMTP mail correspondent password.

The process that handles mail arriving from the network (Internet) is started by the LISTNER process, which listens for incoming service requests on well-known ports. the well-known port number for SMTP services is TCP port 25. This port number is set up in the PORTCONF configuration file for the LISTNER process by adding a entry for SMTP, as described in Configuration Files in the Guardian Environment on page 3-34.

SMTPRCV

When a connection request is received, LISTNER starts another process (called SMTPRCV), which handles requests for a single SMTP connection. The SMTPRCV process creates a new socket and accepts the connection.

The SMTPRCV process establishes a session with TRANSFER, logging in as the gateway mail correspondent. When logging into TRANSFER, SMTPRCV specifies the suffix as the sender of the mail message. This mechanism enables a local recipient to reply to a message when it is received.

The SMTPRCV process verifies the recipient specified through TRANSFER. If the recipient is on another SMTP host, SMTPRCV specifies the gateway correspondent (with a suffix) as the recipient of the message.

SMTPRCV collects the RFC 822 header into a separate attachment. This attachment appears at the end of the message. The subject line appears as a TRANSFER subject line, followed by the body of the message. The header is submitted to TRANSFER at the end of the data.

SMTPSND

A separate process (called SMTPSND) is started by the system administrator as part of the NonStop TCP/IP startup mechanism. The SMTPSND process starts a session with TRANSFER by logging in as SMTPGATE (the gateway correspondent). The SMTPSND process then scans its INBOX for queued-up mail messages. Each message is examined for identifying recipients. If the recipient is SMTPGATE, the real recipient is the user specified in the suffix.

A connection is opened to the host specified for the recipient, and the message is sent to that host. If the host cannot be reached at that time, the message is filed in the RETRYFOLDER. One entry is made in the retry folder for each recipient to whom the mail could not be delivered directly from the INBOX. This mechanism limits retry messages to only those users who have still not received the message.

If the host is unknown, the message is returned to the sender specified by TRANSFER with the notation “ERRORDELIVERING SMTP MAIL” in the subject line. The original message is added as an attachment.

If the INBOX is empty, the RETRYFOLDER is scanned. If the RETRYFOLDER is empty, the SMTPSND waits for 5 minutes (default) or the time specified in the `-`*`time`* parameter when the SMTPSND was started.

If the SMTPSND process runs out of memory, it restarts itself using the name specified in the RUN command (if one is specified). This mechanism allows the SMTPSND process to recover from memory failures without affecting the delivery of any messages.

After changing the configuration file (SMPTCONF), you must restart the SMTP server. Restarting the SMTP server or client does not cause any loss of messages.

The basic intelligence in the SMPTSND and SMTPRCV is the address parsing mechanism. This mechanism is driven from the configuration file $SYSTEM.ZTCPIP.SMTPCONF. This file can parse all legal FC 822 addresses. The system administrator can modify the configuration file to change addresses.

For a detailed description of the operation of the SMTP server and client, refer to the *TCP/IP Applications and Utilities User Guide.* For more detailed technical information on the SMTP, refer to RFC 821 (*DDN Protocol Handbook, Volume 2,* DDN Network Information Center, December, 1985, pp 2-809 through 2-880). Also, refer to the book *Internetworking with TCP/IP* (Douglas E.Comer, Prentice Hall, 1991).

C

Well-Known Port Numbers for TCP and UDP

This appendix lists the port numbers preassigned to specific services when accessed from TCP or UDP.

Table C-1, Table C-2, and Table C-3 give the name or names of each service as used in C programs. These port numbers are provided in the file $SYSTEM.ZTCPIP.SERVICES.

Table C-1. Port Numbers for Network Services

Port Number	Protocol	C Name(s)of Service or Function
7	TCP,UDP	echo
9	UDP	discard, sink null
11	TCP	systat
13	TCP	daytime
15	TCP	netstat
20	TCP	ftp-data
21	TCP	ftp
23	TCP	telnet
25	TCP	smtp, mail
37	TCP,UDP	time, timserver
42	UDP	name, nameserver
43	TCP	whois, nicname (usually tosri-nic)
53	TCP,UDP	domain
101	TCP	hostnames, hostname (usually tosri-nic)
111	TCP,UDP	sunrpc

Table C-2. Port Numbers for Host-Specific Functions

Port Number	Protocol	C Name(s) of Service or Function
69	UDP	tftp
77	TCP	rje
79	TCP	finger
87	TCP	link, ttylink
95	TCP	supdup
105	TCP	csnet-ns

Table C-2. Port Numbers for Host-Specific Functions

Port Number	Protocol	C Name(s) of Service or Function
117	TCP	uucp-path
119	TCP	untp, usenet
123	TCP	ntp
1524	TCP	ingreslock

Table C-3. Port Numbers for UNIX-Specific Services

Port Number	Protocol	C Name(s) of Service or Function
512	TCP	exec
	UDP	biff, comsat
513	TCP	login
	UDP	who, whod
514	TCP	shell, cmd (no passwords used)
	UDP	syslog
515	TCP	printer, spooler (experimental)
517	UDP	talk
520	UDP	route, router, routed
530	TCP	courier, rpc (experimental)
550	UDP	new-rwho, new-who (experimental)
560	UDP	rmonitor, rmonitord (experimental)
561	UDP	monitor (experimental)

D SCF Command Summary

This appendix lists the NonStop TCP/IP SCF commands in alphabetical order.

This quick reference summarizes the commands for those already familiar with how the NonStop TCP/IP SCF commands function. For detailed information about the following commands, refer to Section 4, SCF Reference.

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

```
ABORT [ / OUT file-spec / ] [ ROUTE route-name ]
```

```
ABORT [ / OUT file-spec / ] [ SUBNET subnet-name ]
```

```
ADD [ / OUT file-spec / ] [ ADDRMAP addrmap-name ]

   , IPADDRESS ip-address
   , X121ADDR x25-address
```

```
ADD [ /OUT file-spec/ ] [ ENTRY entry-name ]
          , TYPE { ARP | ATMARP }
          , IPADDRESS  ip addr
          [, MACADDR  mac address ]
          [, ATMADDR atm address ]
          [, PVCNAME "name" ]
          [, SUBNET "name" ]
```

```
ADD [ / OUT file-spec / ] [ ROUTE route-name ]

   { , DESTINATION destination-ip-address
     , GATEWAY     gateway-ip-address    }
   [ , DESTTYPE    { HOST | BROADCAST }   ]
```

```
 ADD [ /OUT file-spec/ ] [ SERVER entry-name ]
          , SUBNET “name”
          , ATMADDR atm address
```

```
ADD [ / OUT file-spec / ]  [ SUBNET subnet-name ]

   , TYPE { LOOPBACK | ATM | ETHERNET | SNAP | X25 }
   , DEVICENAME device-name
   , IPADDRESS  ip-address
   [ , SUNAME   subdevice-name ]
   [ , FORCEQIO { ON | OFF }   ]
   [ , IRDP { ON | OFF }       ]
   [ , ARPSERVER { ON | OFF }
   [ , ATMSEL byte-number ]
```

```
ALTER [ / OUT file-spec / ] [ ADDRMAP addrmap-name ]

   [ , IPADDRESS ip-address ]
   [ , X121ADDR x25-address ]
```

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

   [ [ , TCPSENDSPACE     window-size           ]
   [   , TCPRECVSPACE     window-size           ]
   [   , UDPSENDSPACE     window-size           ]
   [   , UDPRECVSPACE     window-size           ]
   [   , DELAYACKS        { ON | OFF }          ]
   [   , DELAYACKSTIME    delayacks-time        ]
   [   , HOSTNAME         "host-name"           ]
   [   , HOSTID           host-id               ]
   [   , TCPKEEPIDLE      keepidle-time         ]
   [   , TCPKEEPCNT       keepalive-retry-count ]
   [   , TCPKEEPINTVL     keepalive-retry-time  ]
   [   , DEBUG            { ON | OFF }          ]
   [   , FULLDUMP         { ON | OFF }          ]
   [   , ALLNETSARELOCAL  { ON | OFF }       ...]
   [   , TPCOMPAT42       { ON | OFF }
   [   , TCPPATHMTU       { ON | OFF }
   [   , TCPTIMEWAIT      { ON | OFF }
```

```
ALTER [ / OUT file-spec / ] [ SUBNET subnet-name ]

   [ [ , IPADDRESS   ip-address                    ]
     [ , SUBNETMASK  subnet-mask                   ]
     [ , IRDP { ON | OFF }                         ]
     [ , ADDALIAS ip-addr, SUBNETMASK %H0..FFFFFFFF ]
     [ , DELETEALIAS ip-addr                       ]
```

```
DELETE [ / OUT file-spec / ] [ ADDRMAP addrmap-name ]
```

```
 DELETE [ /OUT file-spec/ ] [ ENTRY entry-name ]
        [ , IPADDRESS  ip-addr ]
```

```
DELETE [ / OUT file-spec / ] [ ROUTE route-name ]
```

```
 DELETE [ /OUT file-spec/ ] [ SERVER name ]
```

```
DELETE [/ OUT file-spec / ] [ SUBNET subnet-name ]
```

```
INFO [ / OUT file-spec / ] [ ADDRMAP addrmap-name ]
```

```
INFO [ /OUT file-spec/ ] [ ENTRY entry-name ]
             [ , IPADDRESS  ip addr ]
```

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

   [ , DETAIL ]
```

```
INFO [ / OUT file-spec / ] [ ROUTE route-name ]
```

```
 INFO [ /OUT file-spec/ ] [ SERVER server-name ]
```

```
INFO [ / OUT file-spec / ] [ SUBNET subnet-name ]
     [, DETAIL ]
```

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

   [ , DETAIL ]
```

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

```
NAMES [ / OUT file-spec / ] [ ROUTE route-name ]
```

```
NAMES [ / OUT file-spec / ] [ SUBNET subnet-name ]
```

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

   , CPU cpu-number
```

```
START [ / OUT file-spec / ] [ ROUTE route-name ]
```

```
START [ / OUT file-spec / ] [ SUBNET subnet-name ]
```

```
STATS [ / OUT file-spec / ] [ ADDRMAP addrmap-name ]

   [ , RESET ]
```

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

   [ , RESET ]
```

```
STATS [ / OUT file-spec / ] [ ROUTE route-name  ]
      [, RESET ]
```

```
STATS [ / OUT file-spec / ]  SUBNET subnet-name
      [ , RESET ]      [ , DETAIL ]
```

```
 STATUS [  / OUT file-spec /  ] [ ENTRY entry-name ]
          [ , IPADDRESS  ip-addr ]
          [ , ENDPOINT endpoint-value ]
```

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

```
STATUS [ / OUT file spec / ] [ ROUTE route-name ]
```

```
STATUS [ /OUT file-spec/ ] [ SERVER server-name ]
```

```
STATUS [ / OUT file spec / ] [ SUBNET subnet-name ]
```

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

```
STOP  [ / OUT file-spec / ] [ ROUTE route-name ]
```

```
STOP  [ / OUT file-spec / ] [ SUBNET subnet-name ]
```

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

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

```
TRACE [ / OUT file-spec / ] [ SUBNET subnet-name ]

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

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

   [ , DETAIL ]
```

E SCF Error Messages

This appendix describes the NonStop TCP/IP subsystem SCF error messages.

For the operator display of event messages, see the *Operator Messages Manual*.

TCPIP 00001

```
TCPIP 00001  Invalid file name.
```

Cause. You specified a file with an invalid format.

Effect. The command is not executed.

Recovery. Verify the file-name format and retry the command.

TCPIP 00002

```
TCPIP 00002  INTERNAL ERROR: Case value out of range.
```

Cause. An invalid case value was generated, with no associated case label.

Effect. The SCF command you entered is not executed.

Recovery. Send complete error information to your HP analyst for analysis.

TCPIP 00003

```
TCPIP 00003  Missing Attribute.
```

Cause. You omitted a required attribute from the command.

Effect. The command is not executed.

Recovery. Verify that the required attribute has been included and retry the command.

TCPIP 00004

```
TCPIP 00004  Duplicate Attribute.
```

Cause. You specified the same attribute twice in a command.

Effect. The command is not executed.

Recovery. Omit the duplicate attribute and retry the command.

TCPIP 00005

```
TCPIP 00005  Attribute value out of range attribute-name.
```

attribute-name

is the name of the attribute you specified in an ALTER PROCESS command.

Cause. You specified a value for the ALTER PROCESS command that is outside the valid range.

Effect. The command is not executed.

Recovery. Enter a valid range for the command and retry it. Refer to the ALTER command in Section 4, SCF Reference, for more information on valid ranges.

TCPIP 00007

```
TCPIP 00007  Duplicate address.
```

Cause. The IP address you specified in the ADD SUBNET or ALTER SUBNET command is already being used by another interface.

Effect. The command is not executed.

Recovery. Specify a different IP address and retry the command.

TCPIP 00008

```
TCPIP 00008  Gateway Network Unreachable.
```

Cause. The gateway you specified in the ADD ROUTE command is unavailable.

Effect. The command is not executed.

Recovery. Specify an available gateway and retry the command.

TCPIP 00009

```
TCPIP 00009  Filesystem error.
```

Cause. A file-system error occurred.

Effect. The command is not executed.

Recovery. Take an action based on the file system error received.

TCPIP 00010

```
TCPIP 00010  SNAP MTU not available.
```

Cause. TCP/IP cannot communicate with the manager process to obtain the MTU size.

Effect. The command is not executed.

Recovery. Check or start the manager process.

TCPIP 00011

```
TCPIP 00011  Invalid IP address.
```

Cause. The IP address is invalid.

Effect. The command is not executed.

Recovery. Use a correct IP address.

TCPIP 00012

```
TCPIP 00012  Invalid CPU number.
```

Cause. The processor number is invalid.

Effect. The command is not executed.

Recovery. Use a correct processor number.

TCPIP 00013

```
TCPIP 00013  CPU is already a primary CPU.
```

Cause. The processor number is the primary processor number.

Effect. The command is not executed.

Recovery. Use the backup CPU number.

TCPIP 00014

```
TCPIP 00014  RECSIZE must be at least 300 bytes.
```

Cause. The record size given, or implied, in a TRACE command is too small. It must be at least 300 bytes. The minimum trace record size for TCP/IP has expanded to accommodate additional trace information kept by the NonStop TCP/IP subsystem.

Effect. The command is not executed.

Recovery. Use the RECSIZE parameter while starting a trace. When a larger trace record size is used, there is less chance of trace records being truncated.

TCPIP 00019

```
TCPIP 00019  Primary not allowed, some subnets still in
STARTED state.
```

Cause. This occurs when a Primary command is rejected because at least one subnet is still in the started state and switching to another CPU. This takes any started subnets out of service.

Effect. The command is not executed.

Recovery. Try moving the LAN access interfaces to TCP/IP subsystem backup CPU first.

TCPIP 00020

```
TCPIP 00020  Device access not available from same CPU as
TCPIP.
```

Cause. Device selected for subnet not available between the same CPU pair as the TCP/IP process.

Effect. The command is not executed.

Recovery. Select a device that is available between the same CPU pair as the TCP/IP process or stop the TCP/IP process and restart it specifying a CPU pair that has access to the device.

TCPIP 00021

```
Device access not available from any CPU.
```

Cause. Physical access to the device does not exist. Either the device is not installed or a failure has occurred.

Effect. The command is not executed.

Recovery. Select a device that is available or correct the problem.

TCPIP 00022

```
Unknown LIF device name.
```

Cause. Incorrect device name specified.

Effect. The command is not executed.

Recovery. Select a device that is available or correct the problem.

TCPIP 00023

```
The Device selected for the subnet returned a NULL MAC
address.
```

Cause. The device selected for the subnet returned a NULL MAC address.

Effect. The command is not executed.

Recovery. Try the operation again.

TCPIP 00024

```
Invalid ATM address
```

Cause. ATM address is invalid.

Effect. The command is not executed.

Recovery. Use a correct ATM address.

TCPIP 00025

```
Invalid MAC address.
```

Cause. MAC address is invalid.

Effect. The command is not executed.

Recovery. Use a correct MAC address.

TCPIP 00026

```
ATM API error
```

Cause. A failure was detected in opening an ATM interface.

Effect. The command is not executed.

Recovery. Take an action based on the error and command that failed.

TCPIP 00027

```
ATM adapter not configured in same CPU pair as TCPIP.
```

Cause. ATM adapter is not configured in same CPU pair as TCP/IP.

Effect. The command is not executed.

Recovery. Restart TCP/IP in the same CPU pair as the ATM adapter.

TCPIP 00028

```
ATM selector value already in use.
```

Cause. A subnet is already added using the same ATM adapter and selector value.

Effect. The command is not executed.

Recovery. Add the subnet with a different ATM selector value.

TCPIP 00029

```
Subnet of IP address must be the same as the selected subnet.
```

Cause. The IP address of the ATMARP entry is not on the same subnet as the selected subnet.

Effect. The command is not executed.

Recovery. Verify that the IP address is correct or select a different subnet.

TCPIP 00030

```
Subnet does not exist.
```

Cause. The subnet selected does not exist.

Effect. The command is not executed.

Recovery. Use a correct subnet name or add the subnet and retry the request.

F

NonStop Systems Used as Internet Gateways

Although networks in the Internet are normally connected to each other by industry-standard routers, these networks can be connected by devices called gateways. A gateway is a special-purpose, dedicated computer that attaches to two or more networks and routes packets from one to the other. Gateways route packets to other gateways until the packets can be delivered to the final destination directly across one physical network. An HP NonStop system can function as a gateway.

Figure F-1 shows the relationship between hosts and gateways in the Internet. (A gateway is often a host as well, but not all hosts are gateways.) The Internet addresses shown for the networks and hosts are represented externally as a.b.c.d, where a, b, c, and d are the values of the four octets (*a* being the most significant octet).

In this figure two networks, NETA and NETB, are connected by means of a gateway, HOST4. The three machines labelled HOST1, HOST2, and HOST3 belong to NETA. The seven machines labelled HOST6, HOST7, HOST8, HOST9, HOST10, HOST11, and HOST12 belong to NETB. The gateway, HOST4, belongs to both networks, NETA and NETB.

Figure F-1. Hosts and a Gateway in an Internet

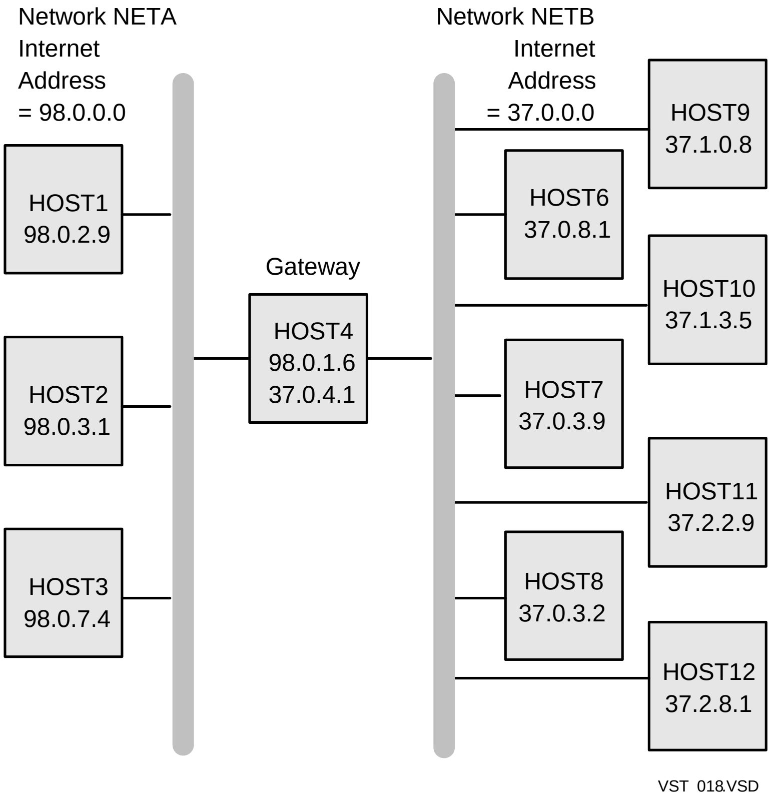
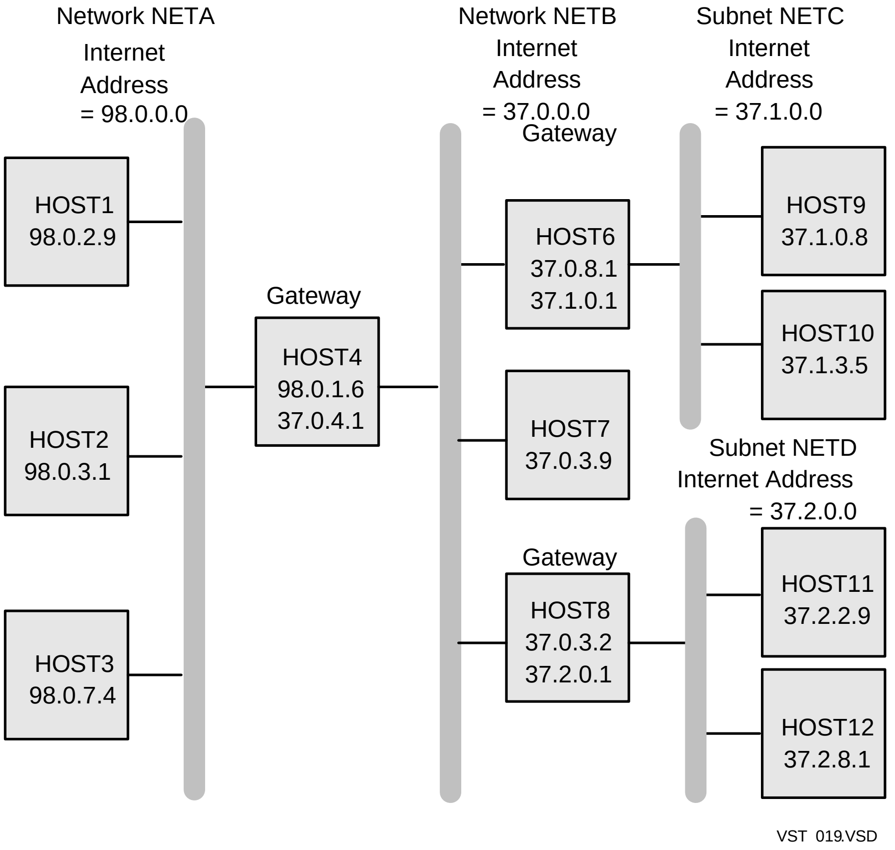


Figure F-2 shows the same two networks illustrated in Figure F-1, but now part of NETB consists of subnets NETC and NETD. Figure F-2 shows the configuration as seen from within NETB. However, to the hosts in NETA, all hosts in NETB, NETC, and NETD appear simply as part of NETB.

Figure F-2. Subnets

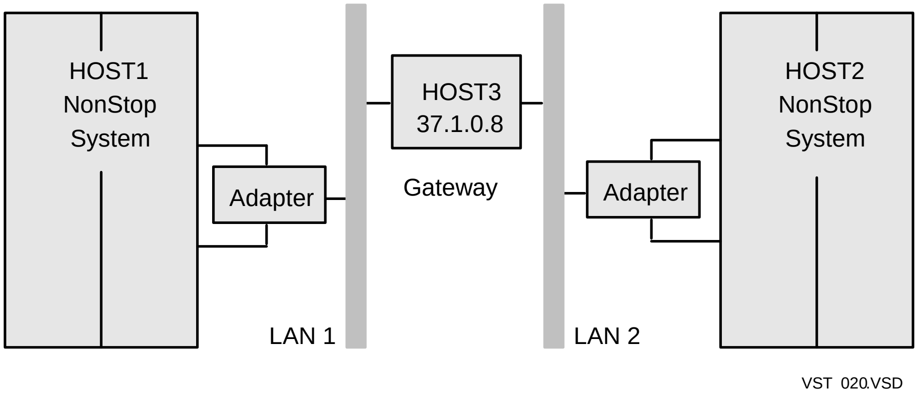


Note. A subnet is associated with a FILTER object in SLSA, which is opened by the NonStop TCP/IP process, or by using the loopback driver for testing. Up to 13 subnets can be attached to each NonStop TCP/IP process: 8 LAN, 1 loopback driver, and 4 X.25 type subnets. A maximum of 4 subnets may be configured for each LAN subnet type.

Figure F-3 shows how a NonStop system can be connected as a host in an Internet. In this example, HOST1 and HOST2 are both NonStop systems. Each NonStop system is connected to a LAN by means of a communications controller. The gateway, HOST3, is part of the same inter-network environment as both HOST1 and HOST2.

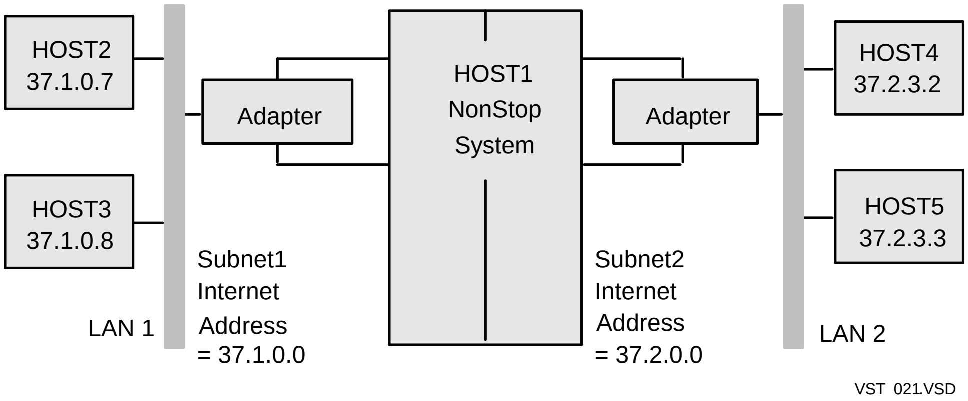
You can set up the Internet so that HOST1 and HOST3 communicate through a number of intermediate gateways, using different communications media and protocols.

Figure F-3. NonStop Systems in an Internet



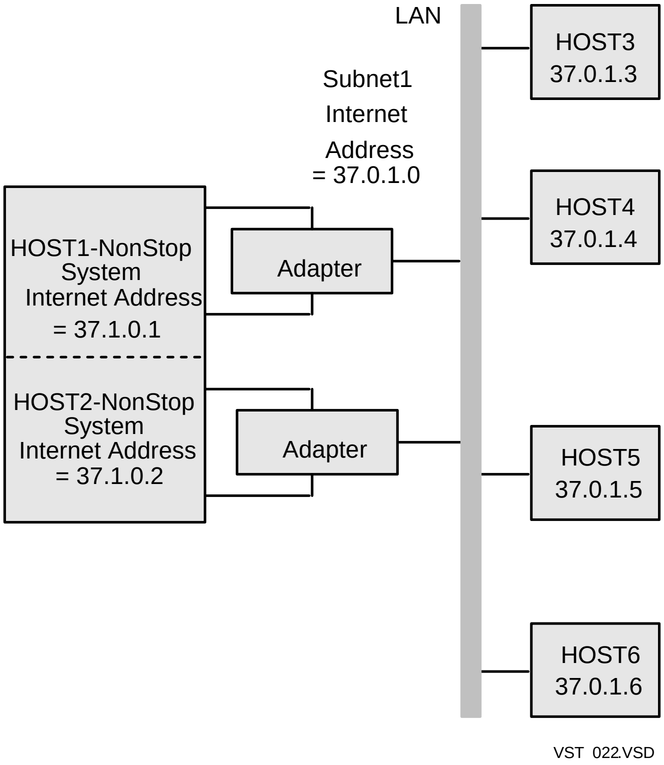
A NonStop system connected as a host in an Internet likewise can act as a gateway between two subnets, as shown in Figure F-4. A single NonStop TCP/IP process runs in the NonStop system, which appears as a single host. In this example, the subnets are on different LANs.

Figure F-4. A NonStop System as a Gateway Between Subnets



In Figure F-5, the NonStop system appears as two separate logical hosts, that have different Internet addresses, connected to a single subnet. Each logical host has its own NonStop TCP/IP process.

Figure F-5. A NonStop System as Two Logical Hosts Connected to a Subnet



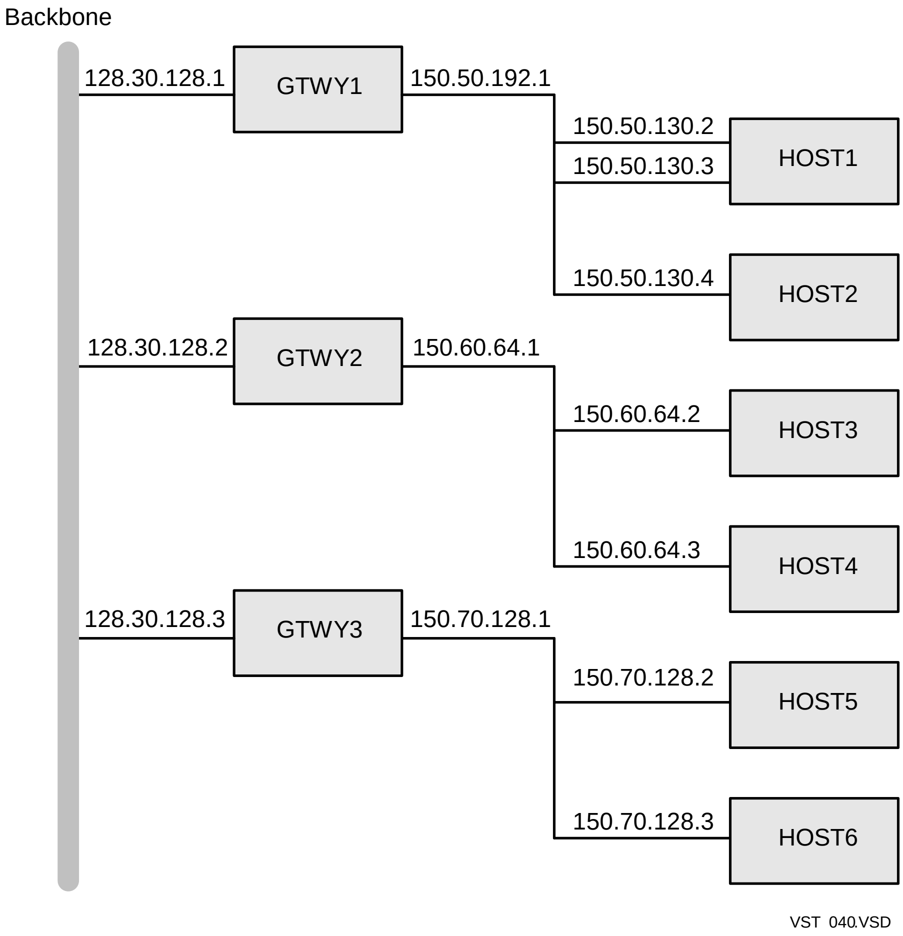
Sample Configuration

This sample configuration provides configuration files for two NonStop systems integrated into the NonStop TCP/IP environment shown in Figure F-6. The environment consists of a backbone communications link with three NonStop gateway systems leading to three different networks. The gateway systems are GTWY1, GTWY2, and GTWY3 They allow communication between six NonStop hosts; HOST1, HOST2, HOST3, HOST4, HOST5 and HOST6.

The sample configuration first shows the configuration files for the NonStop host named HOST1, and then shows the files for the NonStop gateway system (GTWY1) to which it connects.

The IP addresses are Class B addresses. Network routing decisions are therefore based upon the first two octets of the IP address. The sample configuration does not implement subnet addressing. Thus, both systems use the default subnet mask, %HFFFF0000. For information on subnet addressing, see Configuration 2: Startup Files for a Host in a Subnet Addressing Environment on page 3-8 and Appendix B, NonStop TCP/IP Processes and Protocols.

Figure F-6. Three Gateways in the NonStop TCP/IP Environment



Startup Files for HOST1

HOST1 is a NonStop system that has two Ethernet 4 ServerNet Adapters (E4SAs). (For more information about configuring E4SAs, see the *LAN Configuration and Management Manual.*) The two E4SAs attached to the same network appear as two separate hosts to the rest of the network. Although they could be configured using the same NonStop TCP/IP process ($ZTC0), this sample configuration associates them with two separate NonStop TCP/IP processes ($ZTC0 and $ZTC1).

The TCPIPUP2 File

The following TACL command file starts the processes, adds and starts subsystem objects through SCF, and sets appropriate parameters. To add comments, use the word “comment” or a double equal sign (==). Lines which call other files are discussed separately below.

Note. The default NonStop TCP/IP process is $ZTC0 so you do not need to specify defines and params for NonStop TCP/IP applications such as LISTNER and TELSERV for the first part of Example F-1.

Note. The TCP/IP primary and backup processes must be configured in CPUs that have access to the SAC. See Hint on page 1-7.

Example F-1. TCPIPUP2 for Host 1

```
comment  ==== TCPIPUP2 =========TCPIPUP2 ========
comment  TACL command file to bring up NonStop TCP/IP
comment  subsystem
comment  Use HOSTS file for name resolution; not DNS
         ADD DEFINE =TCPIP^HOST^FILE, FILE &
         $SYSTEM.ZTCPIP.HOSTS
comment  Initialize the NonStop TCP/IP processes
comment  (corresponds to HOST1 in Figure F-6)
         TCPIP/NAME $ZTC0, NOWAIT, CPU 1/0
         TCPIP/NAME $ZTC1, NOWAIT, CPU 1/0
comment  ADD and START SUBNET
         SCF/IN $SYSTEM.TCPIP.SCFSBNT/
comment  Init LISTNER for FTPSERV, ECHOSERV, FINGSERV for $ZTC0
         LISTNER/NAME $LSN0,CPU 4,NOWAIT,PRI 160/3 &
         $SYSTEM.ZTCPIP.PORTCONF
comment  Initialize TELSERV process for $ZTC0
         TELSERV/NAME $ZTN0,CPU 2,NOWAIT,PRI 170/
comment  Set TCPIP^PROCESS^NAME to $ZTC1 prior to invoking
comment  LISTNER and TELSERV
         ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC1
comment  Initialize listner for FTPSERV, ECHOSERV,
comment  and FINGSERV for $ZTC1
         LISTNER/NAME $LSN1,CPU 4, NOWAIT, PRI 160/ &
         $SYSTEM.ZTCPIP.PORTCONF/5
comment  Initialize TELSERV process for $ZTC1
         PARAM TCPIP^PROCESS^NAME $ZTC1
         TELSERV/NAME $ZTN1,CPU 2,NOWAIT,PRI 170/
comment  ====== END OF TCPIPUP2 ==== END OF TCPIPUP2 ==
```

The line:

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

sets the `=TCPIP^HOST^FILE` define to point to the desired HOSTS file. Having this define set informs the DNR to use the HOSTS file to translate host names to IP-addresses. For information about the RESOLVER, see RESCONF Details on page 3-37 and the *TCP/IP and IPX/SPX Programming Manual.*

The RUN command:

```
TCPIP/NAME $ZTC0, NOWAIT, CPU 1/0
```

starts the TCP/IP process named $ZTC0 in processors 1 (primary) and 0 (backup).

The RUN command:

```
TCPIP/NAME $ZTC1, NOWAIT, CPU 1/0
```

starts the TCP/IP process named $ZTC1 in processors 1 (primary) and 0 (backup).

Note that TCP/IP starts at a priority of 200, regardless of the priority specified in the RUN command.

The line:

```
ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC0
```

sets the =TCPIP^PROCESS^NAME parameter to $ZTC0. This action must occur prior to starting the LISTNER and TELSERV processes. That way when each process is started, through interrogation of this parameter, they know which Non/Stop TCP/IP process to use. $ZTC0 is the recommended first name to use for the NonStop TCP/IP process. Successive NonStop TCP/IP process names should be $ZTC1, $ZTC2, and so on.

The RUN command:

```
LISTNER/NAME $LSN1,CPU 4, NOWAIT, PRI 160/ &
$SYSTEM.TCPIP.PORTCONF
```

starts the LISTNER process responsible for starting the ECHO, FINGER, and FTP servers when a client request is received by the LISTNER process. Consequently, you should run this process at a higher priority. This command also specifies the location of the PORTCONF file used to designate which ports this process is to listen to. This process requires privileged access to some NonStop TCP/IP ports; therefore, always start with a super group ID.

The line:

```
TELSERV/NAME $ZTN0,CPU 2,NOWAIT,PRI 170/-BACKUPCPU 3
```

starts the Telserv process. When an application (such as Outside View) connects to the IP address of the NonStop TCP/IP process, Telserv presents a window with a Welcome Banner and Service Menu. You can start a TACL using the Service Menu.

The SCFSBNT File

The SCFSBNT file adds and starts subnets and routes. As Example F-2 shows, this file begins by assuming the NonStop TCP/IP process $ZTC0. Thus, the NonStop TCP/IP process must be started in order to add subnets and routes.

Example F-2. SCFSBNT File for TCPIPUP2

```
=== SCFSBNT ===== SCFSBNT ==== SCFSBNT ========
== SCF command file to ADD and START SUBNETs
== This file is created to support HOST1 (Refer to
== Figure F-6)
   ALLOW ALL ERRORS
   ALLOW ALL WARNINGS
== ADD AND START SUBNET $ZTC0.#SN0
   ASSUME PROCESS $ZTC0
   ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN01,IPADDRESS 150.50.130.2
== HELP TCPIP ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
== ADD ROUTES
   ADD ROUTE #ROU0, DESTINATION 128.30.0.0, GATEWAY 150.50.192.1
   ADD ROUTE #ROU1, DESTINATION 150.60.0.0, GATEWAY 150.50.192.1
   ADD ROUTE #ROU2, DESTINATION 150.70.0.0, GATEWAY 150.50.192.1
== HELP TCPIP START SUBNETS & ROUTES
   START SUBNET *
   START ROUTE *

==
== ADD AND START SUBNET $ZTC1.#SN1
==
   ASSUME PROCESS $ZTC1
== HELP TCPIP ADD SUBNET
   ADD SUBNET #SN1,TYPE ETHERNET,DEVICENAME LAN02,IPADDRESS 150.50.130.3
== HELP TCPIP ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
== HELP TCPIP START SUBNET
   START SUBNET *
== HELP TCPIP ADD ROUTE
== ADD ROUTES
   ADD ROUTE #ROU0, DESTINATION 0.0.0.0, GATEWAY 150.50.192.1
== HELP TCPIP START ROUTE
   START ROUTE *
==
=========== END OF SCFSBNT ============= END OF SCFSBNT ===========
```

For the ADD SUBNET command, you can assign the subnet name to be anything you like provided it is no more than seven alphanumeric characters long and begins with an alpha character. The DEVICENAME attribute, which specifies the logical interface (LIF) name associated with the adapter that the NonStop TCP/IP process will access, is required. (See Hint on page 1-7 for determining an appropriate LIF). You can have more than one ServerNet adapter supporting the same NonStop TCP/IP process. You must have a unique IP address for each ServerNet adapter that is physically attached to your network. The IP address links the I/O process name and the NonStop TCP/IP process.

LOOPBACK

When the NonStop TCP/IP process is started, a subnet named #LOOP0 is added automatically. This subnet provides loopback capability without requiring the use of the TCP/IP network. When this #LOOP0 subnet is created, it has an address of 0.0.0.0 in dotted decimal form. For correct operation, the address needs to be changed by using the command:

```
ALTER SUBNET #LOOP0, IPADDRESS 127.1
```

The address 127.1 or 127.0.0.1 is the standard for loopback operation.

ROUTE Objects

A ROUTE object is added for each remote subnet destination with which this host will need to communicate. The Route object specifies the destination network IP address and the gateway IP address to which this host is physically connected.

GATEWAY Attribute

In this sample TCP/IP environment, there are potentially three destination networks to which HOST1 could communicate and thus, three routes, one for each destination network. Notice that in this case, the gateway address is the same for each route. As Figure F-6 shows, HOST1 must route a datagram destined for one of those three networks through GTWY1, which has the IP address of 150.50.192.1. This method is especially useful when you have multiple gateways to multiple networks. When all the routing is through a single gateway, however, there is a simpler way to set up your routing.

Default routing establishes a single route as the default route. This action is particularly useful when you know that most of your TCP/IP traffic is going through a single gateway, as in the case shown in Example F-2. The single route added for the second NonStop TCP/IP process ($ZTC1) in Figure F-6 implements default routing. What indicates that this is a default route is the use of 0.0.0.0 to designate the destination network IP address. You can add more routes for those networks which cannot be reached by using the default route.

Subnet Mask

The subnet mask is not altered in this file. The default mask of %hFFFF0000 is adequate. Because this sample configuration uses a class B IP address on a network without subnets, the first two octets of the IP address are adequate for determining the proper network.

Be aware that adding subnets and routes is not the same as implementing subnetting. This sample configuration configures a host (HOST1) connected to a single network, which is connected to another network through a gateway (GTWY1). Neither of these two networks has implemented subnetting. (For subnettting information. see Configuration 2: Startup Files for a Host in a Subnet Addressing Environment on page 3-8 and Subnet Addresses on page B-5). The use of the SUBNET object simply means implementing TCP/IP within the NonStop environment.

Then, all the subnets and routes of the NonStop TCP/IP processes $ZTC0 and $ZTC1 are started.

The HOSTS File

The HOSTS file is the file used in the absence of a Domain Name Server for resolving the common names of hosts into their corresponding IP addresses. (The HOSTS file in Example F-3 is customized for the purpose of this sample environment.)

Example F-3. HOSTS File for TCPIPUP2

```
########## HOSTS FOR HOST1 ########## HOSTS FOR HOST1 ############
# Filename = \CB1.$SYSTEM.ZTCPIP.HOSTS
# Date     = January 31/93
150.50.192.1  GTWY1B gtwy1b gw1b
128.30.128.1  GTWY1 gtwy1 gw1
128.30.128.2  GTWY2 gtwy2 gw2
128.30.128.3  GTWY3 gtwy3 gw3
127.0.0.1     me loop
150.50.130.2  LAN01 lan01 con1
150.50.130.3  LAN02 lan02 con2
150.50.130.4  HOST2 host2 corp2
150.60.64.1   GTWY2B hgtwy2b gw2b
150.60.64.2   HOST3 host3 corp3
150.60.64.3   HOST4 host4 corp4
150.70.128.1  GTWY3B gtwy3b gw3b
150.70.128.2  HOST5 host5 RD1
150.70.128.3  HOST6 host6 RD2
###########END OF HOSTS ##################END OF HOSTS ###########
```

All text following a pound sign (#) is comment text. Use comment text to note revisions made to the file, the name of the IOP, the hardware address of the ServerNet adapter, and so on. Knowing the hardware address of the ServerNet adapter helps you when you test the network through the HP Tandem Services Manager (TSM).

Begin the IP addresses of the hosts in column one of the HOSTS file. Separate the host name from the address by at least one space. You may have as many aliases as can fit on a single entry line.

The lines in the HOSTS file:

```
127.0.0.1    me loop
150.50.130.2 LAN01 lan01 doc1
150.50.130.3 LAN02 lan02 doc2
```

provide flexibility in testing the environment. When you use the ECHO service to send an echo datagram to `me` or `loop`, you are testing the client and server capabilities of your own ECHO service. If you send an ECHO datagram to lan01 or lan02, you also are testing the actual physical network connection for your HOST1 NonStop TCP/IP environment.

The sample configuration assumes that all the hosts and gateways on this small internet are NonStop hosts. This HOSTS file easily accommodates the IP addresses and names of any host connected to the TCP/IP network.

Startup Files for GTWY1

GTWY1 is a NonStop system that has two E4SA adapters. The configuration for GYWY1 is shown in Figure F-6. Each E4SA adapter is attached to a different network; each has a separate IP address. Since this NonStop system is being configured as a gateway, a single NonStop TCP/IP process interfaces with both I/O processes. If you configure each E4SA adapter with a separate NonStop TCP/IP process, the system could not act as a TCP/IP gateway. Since most of the startup files are very similar to those of HOST1 in Startup Files for HOST1 on page F-5, only the differences between the files will be discussed: the most obvious differences are that there will be a single NonStop TCP/IP process ($ZTC0) and different routes will be added.

The TCPIPUP3 File

As the TCPIPUP3 file in Example F-4 shows, the only significant difference from the file for HOST1 (The TCPIPUP2 File on page F-5) is that a second NonStop TCP/IP process is not started. As a result, there is neither a second LISTNER nor a second TELSERV process.

Note. The NonStop TCP/IP primary and backup processes must be configured in CPU’s that have access to the SAC; (the SAC corresponds to DEVICENAME in Example F-5).

Example F-4. TCPIPUP3 for GTWY1

```
comment  ======== TCPIPUP3 FOR GTWY1=============
comment  Use HOSTS file for name resolution; not DNS
         ADD DEFINE =TCPIP^HOST^FILE, FILE &
$SYSTEM.ZTCPIP.HOSTS
comment  Initialize NonStop TCP/IP process
         TCPIP/NAME $ZTC0, NOWAIT CPU 1/0
comment  ADD and START SUBNET
         SCF/IN $SYSTEM.TCPIP.SCFSBNT/
         ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC0
         LISTNER/NAME $LSN0,CPU 4,NOWAIT,PRI 160/ &
          $SYSTEM.ZTCPIP.PORTCONF/3
         TELSERV/NAME $ZTN0,CPU 2,NOWAIT,PRI 170/
comment  ====== END OF TCPIPUP3 ==== END OF TCPIPUP3 ==
```

The SCFSBNT File

The SCFSBNT file for GTWY1 is different in several respects from that for HOST1. The most notable difference is that there is a single NonStop TCP/IP process ($ZTC0) instead of two processes. So, instead of adding a single subnet to each NonStop TCP/IP process, this sample configuration adds two subnets (one for each adapter) to the single NonStop TCP/IP process.

Note. Both subnets must be added with the GATEWAY ON attribute set.

Example F-5. SCFSBNT File for TCPIPUP3

```
== SCFSBNT FOR GTWY1========= SCFSBNT FOR GTWY1========
== SCF command file to ADD and START SUBNETs
== This file is created to support GTWY1 (Refer to
== Figure F-6)
   ALLOW ALL ERRORS
   ALLOW ALL WARNINGS
== ADD AND START SUBNET $ZTC0.#SN0
   ASSUME PROCESS $ZTC0
== HELP TCPIP ADD SUBNET
   ADD SUBNET #SN0,TYPE ETHERNET,DEVICENAME LAN01,IPADDRESS
150.50.192.1,GATEWAY ON
   ADD SUBNET #SN1,TYPE ETHERNET,DEVICENAME LAN02,IPADDRESS
128.30.128.1,GATEWAY ON
== HELP TCPIP ALTER SUBNET
   ALTER SUBNET #LOOP0, IPADDRESS 127.1
== HELP TCPIP ADD ROUTE
== ADD ROUTES
   ADD ROUTE #ROU1, DESTINATION 150.60.0.0, GATEWAY 128.30.128.2
   ADD ROUTE #ROU2, DESTINATION 150.70.0.0, GATEWAY 128.30.128.3
== HELP TCPIP START SUBNETS & ROUTES
   START SUBNET *
   START ROUTE *
=========== END OF SCFSBNT ============= END OF SCFSBNT ===========
```

The next difference concerns the routes that are added.

```
ADD ROUTE #ROU1, DESTINATION 150.60.0.0, GATEWAY 128.30.128.2
ADD ROUTE  #ROU2, DESTINATION 150.70.0.0, GATEWAY 128.30.128.3
```

Thus, if GTWY1 receives an IP datagram destined for the network 150.60.0.0, the gateway routes that datagram to GTWY2. And likewise, if GTWY1 receives an IP datagram destined for the network 150.70.0.0, the gateway routes that datagram to GTWY3.

The HOSTS File

The HOSTS file, shown in Example F-6, contains only one minor change that makes it differ from the file for HOST1. The address, name, and aliases have been changed for HOST1. HOSTS is the main host file for all the hosts, regardless of the configuration. You can include the current host in your host file. Note that we could have done the same in the HOSTS file for HOST1 as well. Communication destined for HOST1 defaults to the line $LAN01, but the other line can also be used by calling HOST1B.

Example F-6. HOSTS File for TCPIPUP3

```
########## HOSTS FOR GTWY1 ########## HOSTS FOR GTWY ############
# Filename = \CB1.$SYSTEM.ZTCPIP.HOSTS
# Date     = January 31/90
150.50.192.1  GTWY1 gtwy1 gw1
128.30.128.1  GTWY1B gtwy1b gw1b
128.30.128.2  GTWY2 gtwy2 gw2
128.30.128.3  GTWY3 gtwy3 gw3
127.0.0.1     me loop
150.50.130.2  HOST1 host1 h1
150.50.130.3  HOST1B host1b h1b
150.50.130.4  HOST2 host2 corp2
150.60.64.1   GTWY2B gtwy2b gw2b
150.60.64.2   HOST3 host3 corp3
150.60.64.3   HOST4 host4 corp4
150.70.128.1  GTWY3B gtwy3b gw3b
150.70.128.2  HOST5 host5 RD1
150.70.128.3  HOST6 host6 RD2
###########END OF HOSTS ##################END OF HOSTS ###########
```

The changes to the configuration files from one system to the other are not overly complex. With slight modifications, these files could be duplicated and used on each of the other systems in this first sample NonStop TCP/IP configuration environment.

Glossary

This glossary defines terms used both in this manual and in other NonStop TCP/IP manuals. Both industry-standard terms and NonStop terms are included. Because this is a glossary for NonStop TCP/IP as a whole, not all of the terms listed here appear in this manual.

address mask. A bit mask used to select bits from an Internet address for subnet addressing. The mask is 32 bits long and selects the network portion of the Internet address and one or more bits from the local portion.

address resolution. Conversion of an Internet address into a corresponding physical address. Depending on the underlying network, resolution may require broadcasting on a local network.

Address Resolution Protocol (ARP). The Internet protocol used to dynamically bind a high-level Internet address to a low-level physical hardware address. ARP applies only across a single physical network and is limited to networks that support hardware broadcast.

Advanced Projects Research Agency (ARPA). Former name of DARPA, the government agency that funded the ARPANET and DARPA Internet.

ARP. See Address Resolution Protocol (ARP).

ARPA (Advanced Projects Research Agency). See Advanced Projects Research Agency (ARPA).

ARPANET. A pioneering long-haul network funded by ARPA (later DARPA) and built by Bolt, Baranek, and Newman (BBN). It served as the basis for early networking research as well as a central backbone during the development of the Internet.

asynchronous. A mode of serial-data transmission in which characters are sent at random; there is no timing relationship between the end of one character and the start of the next, that is, the transmission is not synchronized with a separate clock signal. The data contains extra bits: a start bit to signal the beginning of a byte and one or more stop bits to signal the end of the byte. These start and stop bits allow the receiver to determine the correct synchronization.

Asynchronous Transfer Mode (ATM). A transfer mode in which the information is organized into cells.

attribute. I1) For the Subsystem Control Facility (SCF), a characteristic of an entity. For example, two attributes of a process might be its program file and its user ID. An attribute is sometimes called a modifier. (2) In OSM client interfaces, a data item associated with a system or cluster resource. All attributes can be viewed, and some can be modified.

ATM. See Asynchronous Transfer Mode (ATM).

bridge. A router that connects two or more networks and forwards packets among them. Usually, bridges operate at the physical network level. For example, an Ethernet bridge connects two physical Ethernet cables and forwards from one cable to the other exactly those packets that are not local. Bridges differ from repeaters; bridges store and forward complete packets, while repeaters forward electrical signals.

broadcast. A packet delivery system that delivers a copy of a given packet to all hosts that attach to it is said to broadcast the packet. Broadcast may be implemented with hardware or software.

BSD. Berkeley Software Distribution.

Class A. The network number is 1 through 127 (1 octet); that is, the first octet is in the range 1-127. The remaining three octets in the address are used for the subnet number and host number.

Class B. The network number is 128 through 191.255 (2 octets); that is, the first octet is in the range 128-191, the second octet is in the range 0-255. The remaining two octets are used for the subnet number and host number.

Class C. The network number is 192.0.0 through 255.255.255 (3 octets); that is, the first octet is in the range 192-255, the second octet is in the range 0-255, and the third octet is in the range 0-255. The remaining octet is used for the subnet number and host number. The subnet number varies in length. The subnet number's width is typically represented by a bit mask. The rest of the available bits uniquely identify the host connected to the subnetwork. LANs connected by way of a gateway to the Internet get their subnet class from the DCA's NIC (Network Information Center). The address classes of standalone, or entirely private, LANs are administered by the LAN administrator. Typical usage calls for all CLASS A addresses to have private LANs.

Class D. A Class D address is a 4-octet multicast group address. The four high-order bits of the address are always 1110; therefore, the first octet is a number in the range 224 through 239 (%HE0 through %HEF). This means that an Internet can have a total of 268,435,456 multicast groups.

collector. An EMS process that accepts event messages from subsystems and logs them in the event log. See also Event Management Service (EMS). Compare distributor.

command message. A SPI message, containing a command, sent from an application program to a subsystem. See also SPI message. Compare response message or event message.

common definition. In DSM programmatic interfaces, a definition (data declaration) used in several commands, responses, or event messages in an SPI interface to a subsystem. See also definition.

connection. The path between two protocol modules that provides reliable stream delivery service. In the Internet, a connection extends from a TCP module on one machine to a TCP module on another machine.

connectionless service. Characteristic of the packet delivery service offered by most hardware and by the Internet Protocol (IP). The connectionless service treats each packet or datagram as a separate entity that contains the source and destination address. Usually, connectionless services can drop packets or deliver them out of sequence.

critical event. A DSM event that is considered to be crucial to the operation of the system or network. Each subsystem determines which of its events are critical, designating them as such by setting the value of the emphasis token to TRUE. Compare noncritical event.

DARPA (Defense Advanced Projects Research Agency). Formerly called ARPA. The government agency that funded research and experimentation with the ARPANET and DARPA Internet.

data communications standard definitions. In DSM, the set of declarations provided by HP for use in all management programs that manage or retrieve event messages from NonStop data communications subsystems. The names of these definitions start with either ZCOM or ZCMK. See also definition or definition files. Compare SPI standard definitions or EMS standard definitions.

data list. In DSM programmatic interfaces, a group of tokens used to separate response records within an SPI message for a response, or used to enclose a single response record, if the program so requests. A data list consists of a list token that denotes a data list (different from the token that starts an error list or a generic list), followed by a response record and an end-list token. See also response record.

DDN (Defense Data Network). Used to loosely refer to the MILNET, ARPANET, and the TCP/IP protocols they use. More literally, it is the MILNET and associated parts of the Internet that connect military installations.

definition. One of the declarations provided by HP for use in applications that call APS or SPI procedures. These definitions are provided in definition files. See also definition files.

definition files. A set of files containing declarations for use in applications that call SPI procedures. SPI has a standard definition file for the Data Definition Language (DDL) and one for each of the programming languages supporting SPI; the latter files are derived from the DDL definition file. Likewise, each subsystem that has a token-oriented programmatic interface has one definition file for DDL and one for each programming language. Some subsystems—for instance, data communications subsystems—have additional, shared definition files. See also SPI standard definitions, data communications standard definitions, or EMS standard definitions.

Distributed Systems Management. A set of tools used to manage NonStop systems and EXPAND networks. These tools include the VIEWPOINT console application, the Subsystem Control Facility (SCF) for data communications subsystems, the Subsystem Programmatic Interface (SPI), the Event Management Service (EMS), the Distributed Name Service (DNS), and token-oriented programmatic interfaces to the management processes for various NonStop subsystems.

distributor. An EMS process that distributes event messages from event logs to requesting management applications, to Guardian console message destinations, or to a collector on another node. See also connection. Contrast collector.

DNS (Domain Name Server). See Domain Name Server.

Domain Name Server. A method for naming resources. The basic function of the domain name server is to provide information about network objects by answering queries.

Domain. In the Internet, a part of the naming hierarchy. Syntactically, a domain name consists of a sequence of names (labels) separated by periods (dots).

DSM. See Distributed Systems Management.

E4SA. See Ethernet 4 ServerNet adapter (E4SA).

ECHO. The name of a program used in the Internet to test the reachability of destinations by sending them an ICMP echo request and waiting for a reply.

EMS (Event Management Service). See Event Management Service (EMS).

EMS standard definitions. The set of declarations provided by EMS for use in event management, regardless of the subsystem. Any application that retrieves tokens from event messages needs the EMS standard definitions. *See also* definition or definition files. Comparedata communications standard definitions or SPI standard definitions.

error. In DSM interfaces, a condition that causes a command or other operation to fail. Contrast Warning.

error number. In DSM programmatic interfaces, a value that can be assigned to a return token, or to the last field of an error token, to identify an error that occurred. Some error numbers are defined in the data communications (ZCOM) definitions; others are defined by individual subsystems.

error token. In DSM programmatic interfaces, a token in a response message that indicates the reason an error occurred during a programmatic command. NonStop subsystems enclose each error token in an error list, which can also contain additional information about the error. A response record must contain a return token and can also contain error lists to explain the error further. The token code for the error token is ZSPI-TKN-ERROR. Its value is a structure consisting of the subsystem ID and an error number identifying the error. *See also* error list, error number, or return token.

Ethernet. A popular local area network technology invented at the Xerox Corporation Palo Alto Research Center. An Ethernet itself is a passive coaxial cable; the interconnections all contain active components. Ethernet is a best-effort delivery system that uses CSMA/CD technology. Xerox Corporation, Digital Equipment Corporation, and Intel Corporation developed and published the standard for 10 Mbps Ethernet.

Ethernet 4 ServerNet adapter (E4SA). A ServerNet adapter for Ethernet local area networks (LANs) that contains four Ethernet ports.

event. In DSM terms, a significant change in some condition in the system or network. Events can be operational errors, notifications of limits exceeded, requests for action needed, and so on.

event log. A file or set of files maintained by EMS to store event messages generated by subsystems.

Event Management Service (EMS). A part of DSM used to provide event collection, event logging, and event distribution facilities. It provides for different event descriptions for interactive and programmatic interfaces, lets an operator or application select specific event-message data, and allows for the flexible distribution of event messages within a system or network. EMS has an SPI-based programmatic interface for both reporting and retrieving events. See also DSM or event message.

event management. The reporting and logging of events, the distribution and retrieval of information concerning those events, and the actions taken by operations personnel or software in response to the events.

event message. A special kind of SPI message that describes an event occurring in the system or network. Compare command message.

Expand. The Expand subsystem enables you to link together as many as 255 geographically dispersed NonStop systems to create a network with the same reliability, capacity to preserve data integrity, and potential expansion as a single NonStop system.

extensible structure. In DSM programmatic interfaces, a structure declared for the value of an extensible structured token. *See also* extensible structured token. Compare fixed structure.

extensible structured token. In DSM programmatic interfaces, a token consisting of a token code and a value that is an extensible structure. Extensible structures can be extended by adding new fields at the end in later RVUs. Such structures are typically used to indicate the attributes of an object being operated on and to return status and statistics information in responses; they can also be used for other purposes. The token is referenced by a token map that describes the structure to SPI so that SPI can provide compatibility between different versions of the structure.

fabric. A simplified way of representing a complex set of interconnections through which there can be multiple and (to the user) unknown paths from point to point. The term fabric is used to refer to the X or Y portion of the ServerNet system area network (ServerNet SAN), for example the X fabric.

Fast Ethernet ServerNet adapter (FESA). The FESA supports one Ethernet 10 Base-T or 100 Base-TX connection and communicates with multiple processors through its dual ServerNet interfaces to the ServerNet fabrics.

FDDI. See Fiber Distribution Data Interface (FDDI).

FESA. See Fast Ethernet ServerNet adapter (FESA).

Fiber Distribution Data Interface (FDDI). An emerging standard for a network technology based on fiber optics. FDDI specifies a 100-mbps data rate using 1300-nanometer light wavelength, and limits networks to approximately 200 km in length, with repeaters every 2 km or less. The access control mechanism uses token-ring technology.

File Transfer Protocol (FTP). The Internet standard, high-level protocol for transferring files from one machine to another. Usually implemented as application level programs, FTP uses the TELNET and TCP protocols. The server side requires a client to supply a login identifier and password before it will honor requests.

filter. In EMS, a file containing a list of criteria against which incoming event messages can be compared. The messages are allowed to pass (all criteria met) or not pass (one or more criteria failed). In the ServerNet LAN Systems Access (SLSA) subsystem (for NonStop S-series systems), filters are logical entities which allow frames received from the LAN to be sorted and delivered to a client. In the SLSA subsystem, filters replace the PORT objects used in K-series systems in the sense that filters are the final destination for data received from the LAN.

FINGER. A protocol providing a method for retrieving status information about one or all of the users on a particular system.

fixed structure. In DSM programmatic interfaces, a multifield structure declared for the value of a simple token. Fields cannot be added to fixed structures in later RVUs. Compare extensible structure.

forwarding distributor. An EMS distributor process that sends selected event messages to an EMS collector on another network node. See also distributor.

FTP (File Transfer Protocol). See File Transfer Protocol (FTP).

full-duplex mode. The communication mode in which data can be transferred in both directions simultaneously. In the Session Layer, no data token is needed.

gateway. A special-purpose, dedicated computer that attaches to two or more networks and routes packets from one to the other. In particular, an Internet gateway routes IP datagrams among the networks to which is connected. The term is loosely applied to any machine that transfers information from one network to another, as in “mail gateway.”

Gateway to Gateway Protocol. The protocol core gateways used to exchange routing information, GGP implements a distributed shortest path routing computation. Under normal circumstances, all GGP participants reach a steady state in which the routing information at all gateways agrees.

G4SA. See Gigabit Ethernet 4-Port ServerNet adapter (G4SA).

GESA. See Gigabit Ethernet ServerNet adapter (GESA).

GGP. See Gateway to Gateway Protocol.

Gigabit Ethernet 4-Port ServerNet adapter (G4SA). A multiport ServerNet adapter that provides 1000 megabits/second (Mbps) data transfer rates between HP NonStop S-series systems, Integrity NonStop servers and Ethernet LANs. The G4SA is the only LAN adapter supported for the I/O Adapter Module (IOAM) enclosure, and it is installed in slots 1, 2, 3, 4, and 5 of an IOAM. Although the G4SA supersedes the Ethernet 4 ServerNet adapter (E4SA), Fast Ethernet ServerNet adapter (FESA), and the Gigabit Ethernet ServerNet adapter (GESA), it cannot be installed in an HP NonStop S-series enclosure.

Gigabit Ethernet ServerNet adapter (GESA). The GESA is a CRU that supports one Ethernet connection and communicates with multiple processors through its dual ServerNet interfaces to the ServerNet fabrics. The adapter exists in copper and fiber versions. The copper version allows switchable or negotiated speeds; the fiber version operates only at 1000 Mbps.

half-duplex mode. The communications mode in which data can be transferred in both directions, but only in one direction at a time, and in which the direction of data flow alternates. In the Session Layer, the data token indicates which side can send data.

header. The initial part of an SPI message. The first word of this header always contains the value -28; the remainder of the header contains descriptive information about the SPI message, most of which is accessible as header tokens. The tokens in an SPI message header differ according to the type of message: the header of a message that contains a command or response differs somewhat from the header of an event message. An application can use SSGET or EMSGET calls to retrieve the values of header tokens, and can use SSPUT calls to change the values of some tokens. However, there are certain basic differences between header tokens and other tokens. See also header token.

header token. In an SPI message, a token that provides information pertaining to the message as a whole. Header tokens differ from other tokens in several ways: they exist in the buffer at initialization and their values are usually set by SSINIT, they can occur only once in a buffer, they are never enclosed in a list, they cannot be moved to another buffer with SSMOVE, and programs cannot position to them or retrieve their values using the NEXTCODE or NEXTTOKEN operation. Programs retrieve the values of header tokens by passing appropriate token codes to SSGET and can change the values of some header tokens by passing their token codes to SSPUT.

Examples of header tokens for commands are the command number, the object type, the maximum-response token, the server-version token, the maximum-field-version token, and the checksum token. Command and response messages contain a specified set of header tokens; event messages, a different set with some overlap. See also SPI message.

hop count. A measure of distance between two points in the Internet. A hop count of n means that *n* gateways separate the source destination.

ICMP. See Interior Gateway Protocol (IGP).

IEEE. See Institute of Electrical and Electronics Engineers (IEEE).

IEEE 802.3. A local area network protocol suite commonly known as Ethernet. Ethernet has either a 10Mbps or 100Mbps throughput and uses Carrier Sense Multiple Access bus with Collision Detection (CSMA/CD. This method allows users to share the network cable. However, only one station can use the cable at a time. A variety of physical medium dependent protocols are supported.

IEE 802.5. A local area network protocol suite commonly known as token ring. A standard originated by IBM for a token-passing ring network that can be configured in a star topology. Versions supported are 4Mbps and 16 Mbps.

IEN. See Internet Engineering Note (IEN).

IGP. See Interior Gateway Protocol (IGP).

Interior Gateway Protocol (IGP). The generic term applied to any protocol used to propagate network reachability and routing information within an autonomous system. Although no standard Internet IGP exists, RIP is among the most popular.

Institute of Electrical and Electronics Engineers (IEEE). An international industry group that develops standards for many areas of electrical engineering and computers.

interactive command. In DSM, a command entered by a human operator rather than by a program. See also programmatic command.

International Organization for Standardization (ISO). A United Nations organization, established to promote the development of standards to facilitate the international exchange of goods and services and to develop mutual cooperation in areas of intellectual, scientific, technological, and economic activity.

Internet. Physically, a collection of packet switching networks interconnected by gateways, along with protocols that allow them to function logically as a single, large, virtual network. When written in uppercase, INTERNET refers specifically to the DARPA Internet and the TCP/IP protocols it uses.

Internet address. The 32-bit address assigned to hosts that want to participate in the Internet using TCP/IP. Internet addresses are the abstraction of physical hardware addresses, just as the Internet is an abstraction of physical networks. Actually assigned to the interconnection of a host to a physical network, an Internet address consists of a network portion and a host portion. The partition makes routing efficient.

Internet Control Message Protocol (ICMP). An integral part of the Internet Protocol (IP) that handles error and control messages. Specifically, gateways and hosts use ICMP to send reports of problems about datagrams back to the original source that sent the datagram. ICMP also includes an echo request/reply used to test whether a destination is reachable and responding.

Internet Engineering Note (IEN). A series of notes developed in parallel to RFCs and available across the Internet from the INIC. IENs contain many of the early theories on the Internet.

Internet Protocol. The Internet standard protocol that defines the Internet datagram as the unit of information passed across the Internet, and that provides the basis for the Internet, connectionless, best-effort, packet-delivery service.

interoperability. The ability of software and hardware on multiple machines from multiple vendors to communicate meaningfully.

IOP. Input/output process. An input/output process (IOP) is a privileged process, residing in a NonStop system processor, which provides an application access to a communications line.

IP. See Internet Protocol.

IP datagram. The basic unit of information passed across the Internet. An IP datagram is to the Internet as a hardware packet is to a physical network. It contains source and destination addresses, along with data.

ISO. See International Organization for Standardization (ISO).

LAN (local area network). Any physical network technology that operates at high speed (usually tens of megabits per second through several gigabits per second) over short distances (up to a few thousand meters).

LANMAN. See LAN manager (LANMAN) process.

LAN. See local area network (LAN).

LAN manager (LANMAN) process. The process provided as part of the ServerNet local area network (LAN) systems access (SLSA) subsystem that starts and manages the SLSA subsystem objects and the LAN monitor (LANMON) process and assigns ownership of Ethernet adapters to the LANMON processes in the system. Subsystem Control Facility (SCF) commands are directed to the LANMON processes for configuring and managing the SLSA subsystem and the Ethernet adapters.

LANMON. See LAN monitor (LANMON) process.

LAN monitor (LANMON) process. The process provided as part of the ServerNet local area network (LAN) systems access (SLSA) subsystem that has ownership of the Ethernet adapters controlled by the SLSA subsystem.

LAPB (Link Access Protocol —Balanced). ITU-T standards that define in the Data Link Layer the requirements for X.25 connections over wide area networks (WANs).

Level 2. A reference to LINK LEVEL communication (for example, frame formats) or link-level connections derived from the ISO 7-layer reference model. For long-haul networks, level 2 refers to the communication between a host computer and a network packet switch (for example, HDLC/LAPB). For local area networks, level 2 refers to physical packet transmission. Thus, a level 2 address is a physical hardware address.

Level 3. A reference to NETWORK-level communication derived from the ISO 7-layer reference model. For the Internet, level 3 refers to the IP and IP datagram formats. Thus, a level 3 address is an Internet address.

LIF. See logical interface (LIF).

LLC (Logical Link Control). See Logical Link Control (LLC).

local area network (LAN). A network that is located in a small geographical area and whose communications technology provides a high-bandwidth, low-cost medium to which low-cost nodes can be connected. One or more LANs can be connected to the system such that the LAN users can access the system as if their workstations were connected directly to it.

logical interface (LIF). The interface that allows an application or another process to communicate with data communications hardware.

Logical Link Control (LLC). An IEEE 802.2 standard for the Data Link Layer of the OSI Reference Model that defines both connection-oriented and connectionless standards over LAN networks.

MAC address (Media Access Control Address). A MAC address is a value in the Medium Access Control sublayer of the IEEE/ISO/ANSI LAN architecture, that uniquely identifies an individual station that implements a single point of physical attachment to a LAN.

management applications. In DSM, an application process that opens a management or subsystem process to control a subsystem. This process can issue SPI commands to subsystems and retrieve EMS event messages to assist in the management of a computer system or a network of systems. A management application is a requester to the subsystems to which it sends commands; the subsystems are servers to the management application.

management process. In DSM, a process through which an application issues commands to a subsystem. A management process can be part of a subsystem, or it can be associated with more than one subsystem; in the latter case, the management process is logically part of each of the subsystems. SCP is the management process for all NonStop data communications subsystems that support DSM. See also subsystem.

manager process. In DSM, a subsystem process with which the SCP management process communicates to control a particular data communications subsystem.

MFIOB. See multifunction I/O board (MFIOB).

MILNET (Military Network). Originally part of the ARPANET, MILNET was partitioned in 1984 to make it possible for military installations to have reliable network service, while the ARPANET continues to be used for research. MILNET uses exactly the same hardware and protocol technology as ARPANET, and there are several interconnection points between the two. Thus, under normal circumstances, MILNET sites are part of the Internet.

multicast. A technique that allows copies of a single packet to be passed to a selected subset of all possible destinations. Some hardware (for example, Ethernet) supports multicast by allowing a network interface to belong to one or more multicast groups. Broadcast is a special form of multicast in which the subset of machines selected to receive a copy of a packet consists of the entire set.

multifunction I/O board (MFIOB). A ServerNet adapter that contains ServerNet addressable controllers (SACs) for SCSI and Ethernet; a service processor; ServerNet links to the processor, to the two ServerNet adapter slots, and to one of the ServerNet expansion board (SEB) slots; and provides connections to the serial maintenance bus (SMB), which connects components within an enclosure to the service processor.

NFS (Network File System). A protocol developed by SUN Microsystems that uses IP to allow a set of cooperating computers to access each other's file systems as if they were local. The key advantage of NFS over conventional file transfer protocols is that NFS hides the differences between local and remote files by placing them in the same name space. NFS is used primarily on UNIX systems, but has been implemented for many systems, including personal computers like an IBM PC and Apple Macintosh.

noncritical event. A DSM event not too crucial to system or network operations. Each subsystem determines which of its events are noncritical by setting the value of the emphasis token to FALSE. Compare critical event.

nonsensitive command. A DSM command that can be issued by any user or program allowed access to the target subsystem—that is, a command on which the subsystem imposes no further security restrictions. For NonStop data communications subsystems, the nonsensitive commands are all those that cannot change the state or configuration of objects (usually information commands). Compare sensitive command.

nowait mode. In Guardian file-system operations and in some APS operations, the mode in which the called procedure initiates an I/O operation but does not wait for it to complete before returning control to the caller. In order to make the called procedure wait for the completion of the operation, the application calls a separate procedure. Compare wait mode.

object. (1) In general HP NonStop use, one or more of the devices, lines, processes, and files in a NonStop subsystem; any entity subject to independent reference or control by one or more subsystems. (2) In DSM use, an entity subject to independent reference and control by a subsystem: for example, the disk volume $DATA or the data communications line $X2502. An object typically has a name and a type known to the controlling subsystem.

object-name template. In DSM, a name that stands for more than one object. Such a name includes one or more wild-card characters, such as * and ?. See also wild-card character.

object type. In DSM, the category of objects to which a specific object belongs: for example, a specific disk file might have the object type FILE, and a specific terminal might have the object type SU (subdevice). A subsystem identifies a set of object types by the objects it manages. The SCF interfaces to data communications subsystems use standard keywords to identify the types. The corresponding programmatic interfaces have object-type numbers (represented by symbolic names such as ZCOM-OBJ-SU) suitable for passing to the SPI SSINIT procedure.

open system. Any computer system that adheres to the OSI standards.

Open Systems Interconnection. A set of standards used for the interconnection of heterogeneous computer systems, thus providing universal connectivity.

OSI. See Open Systems Interconnection.

OSI Reference Model. A communications architecture, adopted by the ISO in 1984, that includes seven layers that define the functions involved in communications between two systems, the services required to perform these functions, and the protocols associated with these services.

packet. The unit of data sent across a packet switching network. While some Internet literature uses it to refer specifically to data sent across a physical network, other literature views the Internet as a packet switching network and describes IP datagrams as packets.

Packet Internet Groper (PING). The name of a program used in the Internet to test the reachability of destinations by sending them an ICMP echo request and waiting for a reply. The term has survived the original program and is now used as a verb, as in “please ping host A to see if it is alive.”

packet switching. A technique in which messages are broken into smaller units, called packets, that can be individually addressed and routed through the network. The receiving-end node ascertains whether all the packets are received and in the proper sequence before forwarding the complete message to the addressee.

PDN. See Public Data Network (PDN).

physical interface (PIF). The hardware components that connect a system node to a network.

physical layer. Layer 1 in the OSI Reference Model. This layer establishes the actual physical connection between the network and the computer equipment. Protocols at the Physical Layer include rules for the transmission of bits across the physical medium and rules for connectors and wiring.

PIF. See “physical interface (PIF).”

PING. See PING.

predefined value. A commonly used value—for instance, a value for a token or a field in a token—that is given a name in a set of definition files.

process. A running entity that is managed by the operating system, as opposed to a program, which is a collection of code and data. When a program is taken from a file on a disk and run in a processor, the running entity is called a process.

programmatic command. In DSM, a command issued by a program rather than by a human operator. Compare interactive command.

protocol. A formal description of message formats and the rules two or more machines must follow to exchange those messages. Protocols can describe low level details of machine-to-machine interfaces (for example, the order in which the bits from a byte are sent across a wire), or high-level exchanges between application programs (for example, the way in which two programs transfer a file across the Internet). Most protocols include both intuitive descriptions of the expected interactions as well as more formal specifications using finite state-machine models.

Public Data Network (PDN). A network with data communications services available to any subscriber.

Request for Comments. he name of a series of notes that contain surveys, measurements, ideas, techniques, and observations, as well as proposed and accepted Internet protocol standards. RFCs are edited but not referenced. They are available across the Internet.

response. In DSM use, the information or confirmation supplied (as part of a response message) to an application by a subsystem in response to a DSM command.

response message. An SPI message sent from a subsystem to an application program in reaction to a command message. Compare command message or event message.

response record. In DSM programmatic interfaces, a set of response tokens usually describing the result when a command is performed on one object. Every response record in a response from a subsystem contains a return token; a response record can also contain error lists that include error tokens. A response can consist of multiple response records, spread across one or more response messages. A response record cannot be split between two response messages. If multiple response records are in a response message, each response record is enclosed in a data list. See also data list. Each response record is required to contain a return token. See also return token.

return token. In DSM programmatic interfaces, the token that indicates whether a command was successful and, if not, why it failed. The token code for the return token is ZSPI-TKN-RETCODE. Its value consists of a single integer field. Compare error token.

RFC (Request for Comments). See Request for Comments.

SAC. See ServerNet addressable controller (SAC).

SCF. See Subsystem Control Facility (SCF).

SCP. See Subsystem Control Point (SCP).

sensitive command. In DSM, a command that can be issued only by a restricted set of Guardian users, such as the owner of a subsystem process. For data communications subsystems, the sensitive commands are those that can change the state or configuration of objects, start or stop tracing, or change the values of statistics counters. Compare nonsensitive command.

ServerNet adapter. A customer-replaceable unit (CRU) that connects peripheral devices to the rest of the system through a ServerNet bus interface (SBI). A ServerNet adapter is similar in function to an I/O controller logic board (LB) and backplane interconnect card (BIC) in NonStop K-series servers.

ServerNet addressable controller (SAC). A controller that is uniquely addressable within one or more ServerNet address domains (SADs) through the node ID and address fields in a request packet. A SAC typically is implemented on some portion of a processor multifunction (PMF) customer-replaceable unit (CRU), an I/O multifunction (IOMF) CRU, or a ServerNet adapter.

ServerNet LAN Systems Access (SLSA) subsystem. A subsystem of the NonStop operating system. The SLSA subsystem enables the protocol I/O processes (IOPs) and drivers to access the ServerNet adapters.

ServerNet wide area network (SWAN) concentrator. A data communications peripheral that provides connectivity to a NonStop S-series server. The SWAN concentrator supports both synchronous and asynchronous data over RS-232, RS-449, X.21, and V.35 electrical and physical interfaces.

service. A set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer can perform on behalf of its users, but not how these operations are implemented. A service relates to an interface between two layers: the lower layer is the service provider, and the upper layer is the service user. Compare protocol.

session. For a management application, the period during which an application can issue commands to a subsystem.

Simple Mail Transfer Protocol (SMTP). The Internet standard protocol for transferring electronic mail messages from one machine to another. SMTP specifies how two mail systems interact, and specifies the format of control messages the two mail systems exchange to transfer mail.

simple token. In DSM programmatic interfaces, a token consisting of a token code and a value that is either a single elementary field, such as an integer or a character string, or a fixed (nonextensible) structure. Compare extensible structured token.

SLSA Subsystem. See ServerNet wide area network (SWAN) concentrator on page -14

SMTP. See Simple Mail Transfer Protocol (SMTP).

SNAP. See Subnetwork Access Protocol (SNAP).

SPI. See Subsystem Programmatic Interface (SPI).

SPI buffer. The buffer that contains an SPI message. See also SPI message.

SPI message. In DSM programmatic interfaces, a message specially formatted by the SPI procedures for communication between a management application and a subsystem or between one subsystem and another. An SPI message consists of a collection of tokens. Note that an SPI message is a single block of information sent at one time, as one interprocess message. There are two types of SPI messages, distinguished by different sets of tokens in the header: command and response messages, and event messages.

SPI procedures. In DSM, the set of Guardian procedures used to build and decode buffers for use in system and network management and in certain other applications.

SPI standard definitions. In DSM programmatic interfaces, the set of declarations available for use with the SPI procedures, regardless of the subsystem. There is also a set of subsystem-specific declarations for each subsystem, and some sets of declarations that apply to multiple subsystems. An application using SPI needs the SPI standard definitions and also the subsystem definitions for all subsystems with which it communicates. *See also* definition. Compare data communications standard definitions or EMS standard definitions.

subject token. In event management, a device, process, or other named entity about which a given event message has information.

subnet address. An extension of the Internet addressing scheme that allows a site to use a single Internet address for multiple physical networks. Outside of the site using subnet addressing, routing continues as usual by dividing the destination address into an Internet portion and local portion. Gateways and hosts inside a site using subnet addressing interpret the local portion of the address by dividing it into a physical network portion and host portion.

subnetwork. One or more intermediate systems that provide relaying and through which end open systems may establish network connections.

Subnetwork Access Protocol (SNAP). In order to run the TCP/IP protocol suite over IEEE networks, the Subnetwork Access Protocol (SNAP) defines the interface between the IP layer and the LLC layer. The interface is accomplished through the use of an extension of the LLC header that contains a predefined Service Access Point (SAP) for use in the Source SAP (SSAP) and the Destination SAP (DSAP) fields of the LLC header.

subsystem. (1) The software and/or hardware facilities that provide users with access to a set of communications service. (2) For DSM, a program or set of processes that manages a cohesive set of objects. Each subsystem has a process through which applications can request services by issuing commands defined by that subsystem; in some cases, this process is the entire subsystem. Many subsystems also have interactive interfaces.

Subsystem Control Facility (SCF). A part of DSM, used to provide a common, interactive management interface for configuring, controlling, and collecting information from NonStop data communications products.

Subsystem Control Point (SCP). .n DSM, the management process for all data communications subsystems on NonStop systems. There can be several instances of this process. Applications using SPI send all commands for data communications subsystems to an instance of this process, which in turn sends the commands on to the manager processes of the target subsystems. SCP also processes a few commands itself. It provides security features, version compatibility, support for tracing, and support for applications implemented as NonStop process pairs. See also management process or manager process.

Subsystem ID (SSID). In DSM programmatic interfaces, a data structure that uniquely identifies a subsystem to SPI. It consists of the name of the owner of the subsystem, a subsystem number that identifies that particular subsystem, and a subsystem version number. The subsystem ID is an argument to most of the SPI procedures.

Subsystem Programmatic Interface (SPI). In DSM, a set of procedures and associated definition files used to define common message-based programmatic interfaces for communication between requesters and servers—for instance, in a management application. SPI includes procedures to build and decode specially formatted messages; definition files in Pascal, TAL, C, COBOL85, and TACL for inclusion in programs, macros, and routines using the SPI procedures; and definition files in DDL for programmers writing their own subsystems.

summary state. In DSM interfaces to data communications subsystems, one of the generally defined possible conditions of an object, with respect to the management of that object. A summary state differs from a state in two ways. First, a summary state pertains to the management of an object, whereas a state may convey other kinds of information about the object. Second, summary states are defined the same way for all data communications subsystems a NonStop system, whereas the set of possible states differs from subsystem to subsystem. The management programming interfaces to data communications subsystems refer to summary states rather than to states. Examples of summary states are STARTED, STOPPED, SUSPENDED, and ABORTING.

SWAN concentrator. See ServerNet wide area network (SWAN) concentrator.

symbolic name. In DSM programmatic interfaces, a name used in programs to refer to commonly used values, token codes, token maps, extensible structures, and other related variables for use in management programs.

TCP (Transmission Control Protocol). The Internet standard transport-level protocol that provides the reliable, full-duplex stream service on which many application protocols depend. TCP allows a process on one machine to send a stream of data to a process on another. It is connection-oriented, in the sense that before transmitting data participants must establish a connection. Software implementing TCP usually resides on the operating system and uses the IP protocol to transmit information across the Internet. It is possible to terminate (shut down) one direction of flow across a TCP connection, leaving a one-way (simplex) connection. The Internet protocol suite is often referred to as TCP/IP because TCP is one of the two most fundamental protocols.

TELNET. The Internet standard protocol for remote terminal connection service. TELNET allows a user at one site to interact with remote timesharing systems at another site just as if the user's terminal is connected directly to the remote machine. That is, the user invokes a TELNET application program that connects to a remote machine, prompts for a login id and password, then passes keystrokes from the user's terminal to the remote machine and displays output from the remote machine on the user's terminal.

TFTP. See Trivial File Transfer Protocol (TFTP).

token. In DSM use, a distinguishable unit in a SPI message. Programs place tokens in an SPI buffer using the SSPUT or SSINIT procedures and retrieve them from the buffer with the SSGET procedure. A token has two parts: an identifying code, or token code, and a token value. In command and response messages, a token normally represents a parameter to a command, an item of information in a response, or control information for the subsystem. In event messages, a token normally represents an item of information about an event or about the event message itself. See also header token.

token number. In DSM programmatic interfaces, the number used by a subsystem to identify each DSM token that it defines. The token type and the token number together form the token code.

token ring. 1)þþThe token access procedure used on a network with a sequential or ring topology. (2) A data link level protocol designed to transfer data over ring-oriented LANs. The token ring technique is based on the use of a particular bit pattern called a token that circulates around the ring when all stations are idle.

token type. In DSM programmatic interfaces, the part of a DSM token code that identifies the data type and length of the token value. The token type and the token number together form the token code.

token value. In DSM programmatic interfaces, the value assigned to a DSM token.

Trivial File Transfer Protocol (TFTP). The Internet standard protocol for file transfer with minimal capability and minimal overhead. TFTP depends only on the unreliable, connectionless datagram delivery service (UDP), so it can be used on machines like diskless workstations that keep such software in ROM and use it to bootstrap themselves.

UDP. See User Datagram Protocol (UDP).

User Datagram Protocol (UDP). The Internet standard protocol that allows an application program on one machine to send a datagram to an application program on another machine. UDP uses the Internet Protocol to deliver datagrams. Conceptually, the important difference between UDP and IP is that UDP messages include a protocol port number, allowing the sender to distinguish among multiple destinations (application programs) on the remote machine. In practice, UDP also includes a checksum over the data being sent.

wait mode. In the Guardian operating system, the mode in which the called procedure waits for the completion of an I/O operation before returning a condition code to the caller. Compare nowait mode.

WAN. See wide area network (WAN).

WAN manager process. The WAN manager process starts and manages the WAN subsystem objects including the ConMgr and WANBoot processes.

WAN subsystem. See wide area network (WAN) subsystem.

Warning. In DSM interfaces, a condition encountered in performing a command or other operation, that can be significant but does not cause the command or operation to fail. A warning is less serious than an error. Compare error.

well-known port. Any of a set of protocol ports preassigned for specific uses by transport level protocols (that is, TCP and UDP). Servers follow the well-known port assignments so clients can locate them. Examples of well-known port numbers include ports assigned to echo servers, time servers, remote login (TELNET) servers, and file transfer (FTP) servers.

wide area network (WAN). A network that operates over a larger geographical area than a local area network (LAN)—typically, an area with a radius greater than one kilometer. The elements of a WAN may be separated by distances great enough to require telephone communications. Contrast with local area network (LAN).

wide area network (WAN) subsystem. The Subsystem Control Facility (SCF) subsystem for configuration and management of WAN objects in G-series and H-series RVUs.

wild-card character. A character that stands for any possible character(s) in a search string or in a name applying to multiple objects. In DSM object-name templates, two wild-card characters can appear: ? for a single character and * for zero, one, or more consecutive characters. See also object-name template.

X.25. The CCITT standard protocol for transport-level network service. Originally designed to connect terminals to computers, X.25 provides a reliable stream transmission service that can support remote login.

X.25 Access Method. *See* X25AM.

X25AM (X.25 Access Method). A NonStop product that implements, for WANs, the services of the Network Layer and layers below.

X.25 network. Any network or subnetwork linked using X.25 standards. X.25 standards are CCITT standards that define packet switching carrier communication in the Network Layer over wide area networks (WANs). See also Internet and packet switching.

$ZZLAN. See LAN manager (LANMAN) process.

$ZZWAN. See WAN manager process.

See WAN manager process.

Index