Open System Services NFS SCF Reference Manual

Abstract

This manual describes the use of the Subsystem Control Facility (SCF) to configure and maintain the Compaq *NonStop*™ Kernel Open System Services (OSS) Network File System (NFS). It is written for system managers and operators.

Product Version

OSS NFS G00

Supported Releases

This manual supports function added to the D41.00 release and all subsequent D4x releases and supports all G-series releases until otherwise indicated by its replacement publication.

Part Number	**Published**
522582-001	February 2002

Document History

Part Number	Product Version	Published
114423	OSS NFS D40	May 1996
420036-001	OSS NFS G00	December 1998
522582-001	OSS NFS G00	February 2002

Open System Services NFS SCF Reference Manual

Glossary Index Figures Tables

SCF Commands for OSS NFS (continued)

SCF Commands for OSS NFS (continued)

SCF Commands for OSS NFS (continued)

SCF Commands for OSS NFS (continued)

SCF Commands for OSS NFS (continued)

SCF Commands for OSS NFS (continued)

What's New in This Manual

Manual Information

Abstract

This manual describes the use of the Subsystem Control Facility (SCF) to configure and maintain the Compaq *NonStop*™ Kernel Open System Services (OSS) Network File System (NFS). It is written for system managers and operators.

Product Version

OSS NFS G00

Supported Releases

This manual supports function added to the D41.00 release and all subsequent D4x releases and supports all G-series releases until otherwise indicated by its replacement publication.

Part Number	Published
522582-001	February 2002

Document History

Part Number	Product Version	Published
114423	OSS NFS D40	May 1996
420036-001	OSS NFS G00	December 1998
522582-001	OSS NFS G00	February 2002

New and Changed Information

This manual has been revised to document support for Parallel Library TCP/IP. The manual title has been changed so that the manual is grouped with the other OSS NFS manuals in the Total Information Manager (TIM) display.

About This Manual

This manual describes the Subsystem Control Facility (SCF) interactive interface that allows you to configure, control, and monitor the Open System Services (OSS) implementation of the Network File System (NFS). OSS NFS is a network file server compatible with the Sun Microsystems NFS, version 2, protocol. NFS clients on remote hosts can use OSS NFS servers to store and access files on the Compaq NonStop range of servers.

Intended Audience

Typically, the information provided in this manual is used by operators, system managers, and system administrators who configure and manage the OSS NFS subsystem. The commands described here can also be helpful to NFS users who want information about the files they are using.

How to Use this Manual

This manual contains the following information:

- Section 1, Introduction, provides an overview of the OSS NFS subsystem and detailed descriptions of the objects you control using the SCF commands.
- Section 2, SCF Commands for OSS NFS, describes the syntax of the SCF commands you use to manage the OSS NFS subsystem and provides considerations for using each command and examples.
- Appendix A, SCF Command Summary for OSS NFS, contains a summary of the syntax of SCF commands.
- Appendix B, Summary of All SCF Commands, contains a summary of all SCF commands—general and subsystem-specific.

Where to Go for More Information

This manual provides details about SCF commands as they apply to the OSS NFS subsystem. If you have not used SCF, you should read the *Subsystem Control Facility (SCF) Reference Manual* for D-series information, or the *SCF Reference Manual for G-Series Releases* for G-series information.

Before you can begin to operate the OSS NFS subsystem, you must perform some configuration steps explained in the *Open System Services NFS Management and Operations Guide*.

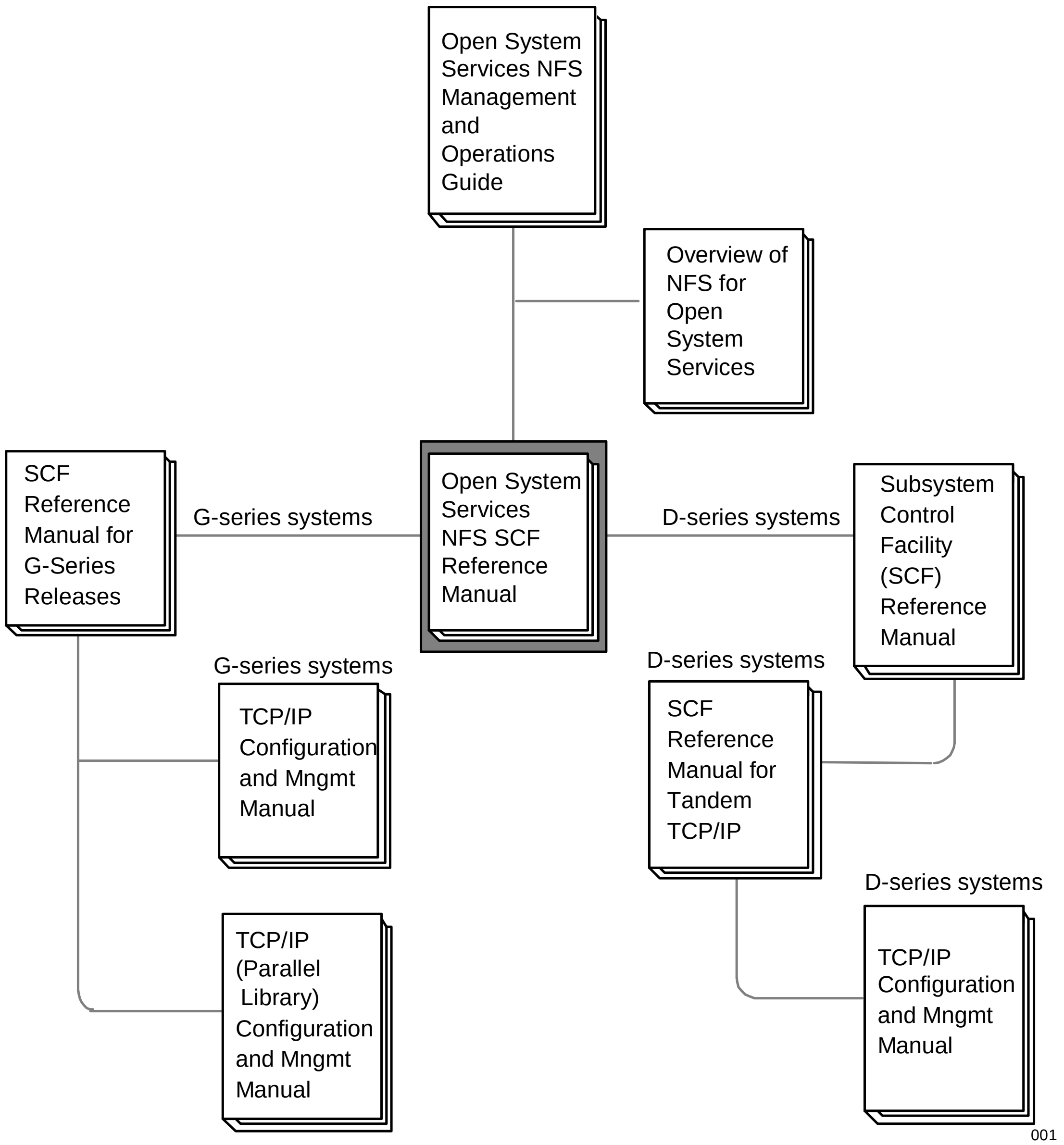
You might also find information in the *PTrace Reference Manual* helpful.

If you need information about the OSS environment, consult the following manuals:

- Open System Services User’s Guide
- Open System Services Porting Guide

- Open System Services Programmer's Guide

The following diagram shows the manuals used to install, configure, manage, and operate OSS NFS. The diagram is organized symmetrically, with programmatic interface manuals on the left and the interactive interface manuals on the right. Note that this manual is highlighted.


Open System Services NFS Management and Operations Guide
Overview of NFS for Open System Services
SCF Reference Manual for G-Series Releases
G-series systems
Open System Services NFS SCF Reference Manual
D-series systems
Subsystem Control Facility (SCF) Reference Manual
G-series systems
TCP/IP Configuration and Mngmt Manual
D-series systems
SCF Reference Manual for Tandem TCP/IP
D-series systems
TCP/IP Configuration and Mngmt Manual
TCP/IP (Parallel Library) Configuration and Mngmt Manual
001

Your Comments Invited

After using this manual, please take a moment to send us your comments. You can do this by returning a Reader Comment Card or by sending an Internet mail message.

A Reader Comment Card is located at the back of printed manuals and as a separate file on the User Documentation disc. You can either fax or mail the card to us. The fax number and mailing address are provided on the card.

Also provided on the Reader Comment Card is an Internet mail address. When you send an Internet mail message to us, we immediately acknowledge receipt of your message. A detailed response to your message is sent as soon as possible. Be sure to include your name, company name, address, and phone number in your message. If your comments are specific to a particular manual, also include the part number and title of the manual.

Many of the improvements you see in manuals are a result of suggestions from our customers. Please take this opportunity to help us improve future manuals.

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

[ ] 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 ]...
```

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 %Hþnotation precedes a hexadecimal number. For example:

```
%005400
```

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

Change Bar Notation

Change bars are used to indicate substantive differences between this edition of the manual and the preceding edition. 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.

1 Introduction

This section contains the following information:

- A general description of the Subsystem Control Facility (SCF), which is an operator interface to the Network File System (NFS) subsystem.
- Descriptions of the OSS NFS object types supported by SCF.
- Descriptions of the OSS NFS *`object-specs`* and naming rules for each object.
- A general description of the attribute specifiers you use to define objects.
- Descriptions of the summary states supported by OSS NFS.

Overview

The OSS NFS subsystem is a network file server that uses the Network File System (NFS) protocol developed by Sun Microsystems. OSS NFS allows NFS client users on workstations, PCs, or other network hosts to create, access, and share files on NonStop Himalaya systems. Clients can share UNIX, DOS, Windows NT, Macintosh, and Enscribe disk files with users and applications on other systems. To use OSS NFS, the client machine must be able to issue NFS client requests using the Transmission Control Protocol/Internet Protocol (TCP/IP) over a local area network (LAN).

This manual describes the subsystem-specific details for using SCF to configure, control, and inquire about the OSS NFS subsystem, which operates through the OSS NFS manager process, the LAN interface process, and NFS server processes.

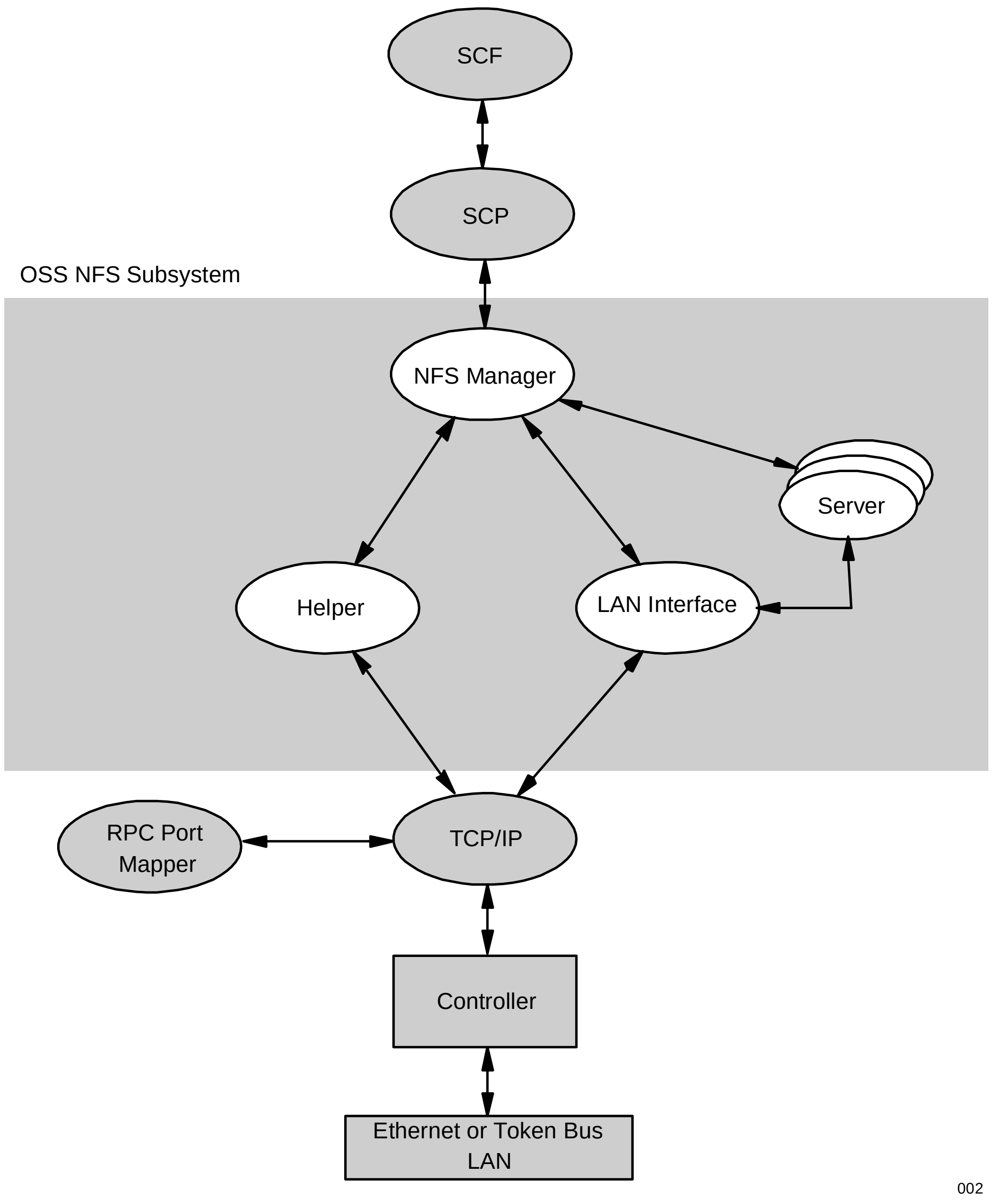
To use SCF, the OSS NFS manager process must first be configured as described in the *Open System Services NFS Management and Operations Guide.*

Figure 1-1, illustrates the relationships among the OSS NFS, Compaq NonStop TCP/IP, and SCF subsystems:

- The Subsystem Control Point (SCP) provides the programmatic interface to OSS NFS using standard Subsystem Programmatic Interface (SPI) commands and responses.
- The NFS manager program (NFSMGR), typically named $ZNFS, runs as a process. NFSMGR accepts commands, reports errors, and generates event messages.
- The helper process (NFSMGR2 program) is started and stopped as needed to provide address resolution and other LAN-related services to the manager process.
- The LAN interface process (NFSLAN program) acts as a front end for NFS requests received from the NonStop TCP/IP subsystem. It forwards mount protocol requests to the manager process and relays NFS protocol requests to the appropriate server processes.
- A server process (NFSSVRHP program) runs for each OSS NFS SERVER object added and started. The server processes manage the files in the hierarchy defined for each server.

- The Remote Procedure Call (RPC) port mapper process $ZPMx[x], where x[x] indicates a TCP/IP stack number, registers services using the RPC protocol. For example, $ZPFM0 runs over $ZTC0.

Figure 1-1. SCF and the OSS NFS Subsystem

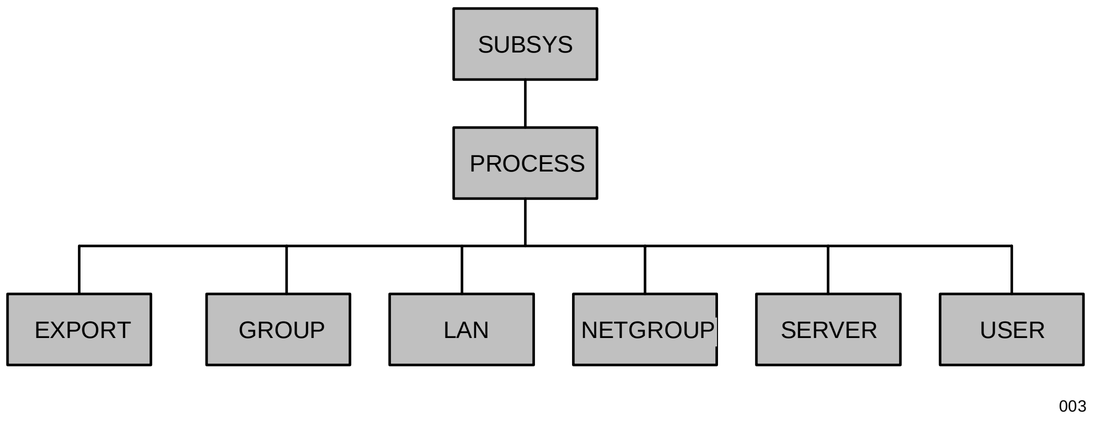


Object Types

Using SCF you can manipulate the OSS NFS subsystem by entering commands that act on one or more objects. Each OSS NFS subsystem object belongs to a particular object type. An object name uniquely identifies a specific object within the subsystem.

Figure 1-2 shows the object types supported by OSS NFS and their hierarchical order. The SUBSYS (subsystem) object is at the highest point in the hierarchy. The PROCESS object is subordinate to the SUBSYS object, and all other objects are peers and are subordinate to the PROCESS object. The hierarchy is important when issuing commands to the OSS NFS subsystem for processing. For example, because the LAN and SERVER objects are subordinate to the PROCESS object, any commands pertaining to a LAN or SERVER can be issued only when the PROCESS object is in the STARTED summary state. (See Summary States on page 1-7.)

Figure 1-2. NFS Object Hierarchy



The description of each object type is as follows:

SUBSYS Identifies the OSS NFS subsystem as a whole. There is only one SUBSYS object for each OSS NFS subsystem. The name of the SUBSYS object is the manager process name. Remote mounts requested by NFS clients are considered to be opens of the SUBSYS object.

PROCESS Identifies the OSS NFS manager process. There is only one PROCESS object for each OSS NFS subsystem. The manager process is responsible for handling SPI requests, and starting, controlling, and stopping all other subsystem processes.

The name of a PROCESS object is the manager process name. The recommended name for the manager process is $ZNFS.

EXPORT	Identifies a server mount point, and all the files managed by that server that are to be exported. Note that although you can define EXPORT objects for non-existent directories or directories that are not server mount points these EXPORT objects are ignored. Export permissions are only checked by the NFS manager process for the mount point of the server responsible for the directory being remotely mounted.
GROUP	Identifies a group of NFS users.
LAN	Identifies the local area network (LAN) interface process that acts as a front end for NFS requests received from the NonStop™ TCP/IP communications process.
NETGROUP	Identifies a group of host machines and users within specified network domains. Netgroups can be used in the access lists attached to EXPORT objects.
SERVER	Identifies a server process. An OSS NFS server provides access to a specific OSS file set. Each server has an attribute that identifies a mount point within the local hierarchy; that is, the hierarchy of the system on which the OSS NFS subsystem is operating. A local mount occurs whenever a server is started.
USER	Identifies an NFS user by a user name and user number.

To control the OSS NFS subsystem, you use the SCF commands documented in Section 2, SCF Commands for OSS NFS. As shown in Section 2, the syntax of each command contains the variable *`object-spec`*, which you use to specify the type and name of the object on which you want the command to operate.

You do not need to specify an object type for any command that supports the NULL object. In OSS NFS, these commands are NAMES and VERSION. The NULL object is not within the object-type hierarchy.

Object-Name Syntax

The syntax and rules for specifying OSS NFS object names vary slightly from the standard SCF object-name syntax. The original SCF object-name design was intended to cover two types of object names: Enscribe file names and simple subsystem-defined object names. To handle NFS path names (which can be up to 1024 characters long and can contain any character except ASCII NUL), this design had to be expanded.

OSS NFS object names are of the form:

`[\system.][$mgr.]objname`

where each of the object-name components are defined as follows:

`\system`

is the name of the system on which the manager process resides. System names follow the standard Compaq NonStop Kernel operating system naming conventions and cannot contain wild-card characters.

`$mgr`

is the name of the NFS manager process (usually $ZNFS). An object name must contain this prefix unless the currently assumed object name (from a previously entered ASSUME command) is the manager process name.

`objname`

is the actual object name of the specific object. The syntax and format of this portion of the object name varies based on the object type. Table 1-1provides a summary of the syntax and format for each OSS NFS object type.

Table 1-1 provides a summary of the naming conventions for each of the object-name components.

Table 1-1. Naming Conventions for OSS NFS Objects

Type of Object	Class of Object Name	Type of Name	Length (in Characters)	Characters Allowed	First Character	Case Sensitive
EXPORT	1	OSS pathname*	1 to 1024**	Any ASCII but NUL***	Any ASCII but NUL	Yes
GROUP	0	NFS group name	1 to 64	Any ASCII but colon (:) and NUL	Letter, number, or underscore (_)	Yes
LAN	0	LAN object name	1 to 8	Alphanumeric	Letter	No
NETGROUP	0	NFS netgroup name	1 to 64	Any ASCII but NUL	Letter, number, or underscore (_)	Yes
PROCESS	0	device name	2 to 6	Alphanumeric	Dollar sign ($) followed by letter	No
SERVER	0	SERVER object name	1 to 8	Alphanumeric	Letter	No
SUBSYS	0	device name	2 to 6	Alphanumeric	Dollar sign ($) followed by letter	No
USER	0	NFS user name	1 to 64	Any ASCII but colon (:) and NUL	Letter, number, or underscore (_)	Yes

* Indicates that the pathname is of the form /[*`file-name`*[/*`file-name`*]...] where *`file-name`* is a case-sensitive name that can include up to 248 ASCII characters, excluding ASCII slash (/) and NUL.

** Indicates that the length is determined after escape characters are removed and octal codes are converted to one character (see Object-Name Templates on page 1-6).

*** Indicates that multiple consecutive slashes (//) are interpreted as one slash (/).

Object-Name Templates

Object-name templates allow you to specify multiple objects by entering either a single wild-card character, or text and one or more wild-card characters. With an object-name template, you can use one object name to specify multiple objects of a given object type.

Wild-Card Characters

You can use wild-card characters to specify object-name templates for all OSS NFS object names except the PROCESS name. You can use the following wild-card characters in object-name templates:

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

- A whole, separated name; for example, SERVER $ZNFS.* selects all servers subordinate to the manager process $ZNFS.
- A trailing string; for example, SERVER $ZNFS.NA* selects all servers subordinate to $ZNFS that have names that start with NA.
- An undefined number of characters; for example, SERVER $ZNFS.R*5 selects all servers subordinate to $ZNFS that start with R and end with 5.
- Use a double asterisk (**) as a wildcard in EXPORT object names.

? Use a question mark to represent a single unknown character in a specific position; for example, SERVER $ZNFS.S?1 selects all servers subordinate to $ZNFS that begin with S, end with 1, and contain only one character between the S and the 1.

If you have set a default PROCESS name by using the ASSUME command, you can omit the PROCESS name and use the asterisk (*) to specify all objects of the specified object type under the assumed process. For example, the next two commands set the default process to $ZNFS and display information about all servers:

```
-> ASSUME PROCESS $ZNFS
-> INFO SERVER *
```

You cannot use wild-card characters in system names or the manager process name.

Escape Characters

In NFS export names, user names, group names, or netgroup names, you must precede the following characters with the escape character, which is a backslash (\), if you want the character to be interpreted as part of the name:

asterisk (*)

question mark (?)

left and right square brackets ([ ])

backslash (\ )

The left square bracket and right square bracket are reserved for future use as wild-card characters.

You can also use the backslash escape character to specify the octal code for an ASCII character; for example, \033.

In addition, if these names include any of the following characters, you must enclose the name in quotes:

space

comma (,)

semicolon (;)

ampersand (&)

Attribute Specifiers

Each OSS NFS object has an associated set of attributes. Using the ADD command you can define the attributes of all objects except the PROCESS and SUBSYS objects. Using the ALTER command you can alter some or all of the attributes of any object. An attribute specifier consists of an attribute name and value. Default values exist for many attributes. You can use the INFO command to display the current attribute values for an object. To rename a file using the DELETE command, you also use an attribute specifier.

The details of the required syntax for each attribute specifier you can use with OSS NFS are described under the individual commands in Section 2, SCF Commands for OSS NFS.

Summary States

Only the SUBSYS, PROCESS, SERVER, and LAN objects have operational states, known as summary states. The summary state of an object at a given instant is important because certain commands have no effect on objects when they are in one state, but can affect the object when it is in another state. If a summary state affects the operation of an SCF command, the effect is noted in the “Considerations” subsection of the command description in Section 2, SCF Commands for OSS NFS.

The summary states supported by the OSS NFS subsystem are STARTING, STARTED, STOPPING, and STOPPED:

STARTING summary state	an object is being initialized and is attempting to start
STARTED summary state	an object is available for service requests
STOPPING summary state	an object is being stopped; no new service requests are accepted
STOPPED summary state	an object is not available for service requests

2 SCF Commands for OSS NFS

This section contains the following information:

- A table describing the Subsystem Control Facility (SCF) commands supported by the OSS NFS subsystem and the object types applicable to each command
- Instructions for entering SCF commands
- The following detailed information about each SCF command that you can use to configure and control the OSS NFS subsystem:
    - A description of the command function
    - The command syntax
    - The *`object-spec`*, which shows the object type and object name
    - Considerations for using the command
    - Descriptions of the *`attribute-specs`*
    - Examples of using the command

Supported Commands and Object Types

This manual describes the SCF commands interpreted specifically for the OSS NFS subsystem. The *Subsystem Control Facility (SCF) Reference Manual* (for D-series releases) and the *SCF Reference Manual for G-Series Releases* provide information about SCF commands generally supported by SCF; for example, the ASSUME, HELP, and ENV commands. You should be familiar with that information before reading the subsystem-specific information provided here. Table B-1in Appendix B shows the subsystem-specific commands and general SCF commands.

Detailed syntax information on the OSS NFS object types and the commands that apply to them is provided when you type the following command:

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

Table 2-1 lists the SCF commands supported by OSS NFS and the object types to which each SCF command applies.

Table 2-1. SCF Commands and Objects for OSS NFS

Command	EXPORT	GROUP	LAN	NET GROUP	PROCESS	SERVER	SUBSYS	USER	NULL
ABORT	No	No	Yes	No	No	Yes	Yes	No	No
ADD	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No
ALLOWOPENS	No	No	No	No	No	No	Yes	No	No
ALTER	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
DELETE	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No
INFO	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
LISTOPENS	No	No	No	No	Yes	No	Yes	No	No
NAMES	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
PRIMARY	No	No	No	No	Yes	No	No	No	No
START	No	No	Yes	No	No	Yes	Yes	No	No
STATS	No	No	Yes	No	Yes	Yes	No	No	No
STATUS	No	No	Yes	No	Yes	Yes	Yes	No	No
STOP	No	No	Yes	No	No	Yes	Yes	No	No
STOPOPENS	No	No	No	No	No	No	Yes	No	No
TRACE	No	No	Yes	No	No	No	No	No	No
VERSION	No	No	Yes	No	Yes	Yes	No	No	Yes

Entering SCF Commands

You start SCF interactively, using the implicit TACL RUN command:

```
>SCF
```

It is seldom necessary to specify SCF RUN parameters because the default values are appropriate for most situations. For a detailed description of the RUN parameters that apply to SCF, refer to the *Subsystem Control Facility (SCF) Reference Manual* (for D-series systems) or the *SCF Reference Manual for G-Series Releases*.

If you work with OSS NFS on a regular basis, it might be convenient to set up an SCFCSTM file containing the following commands to automatically set up your SCF session with OSS NFS:

```
->ASSUME PROCESS $ZNFS
->ALLOW ERRORS
->ALLOW WARNINGS
```

If you only have one or two commands to issue, it might be more convenient to issue them as a single TACL RUN command. For example:

```
>SCF STATUS SUBSYS $ZNFS, SUB ALL, SEL STARTED
>SCF STOP SUBSYS $ZNFS, ORDERLY
```

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

SCF indicates it is ready to process commands by displaying its prompt: the hyphen and greater-than sign (->). Additional information can be added in front of this prompt by using the SETPROMPT command, as described in the *Subsystem Control Facility (SCF) Reference Manual* (for D-series releases) or the *SCF Reference Manual for G-Series Releases*.

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. For example:

```
->ABORT LAN $ZNFS.LAN0
```

The object specifiers (object types, and object names) supported by OSS NFS (including the use of the ASSUME command to set a default object type or object name) are discussed in Section 1, Introduction. In addition, OSS NFS also allows the use of multiple object names with one command.

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. For example, the following command defines the execution priority (PRI attribute) for a LAN object:

```
->ALTER LAN $ZNFS.LAN0, PRI 148
```

You can enter multiple SCF commands at a single prompt by separating the commands with semicolons. For example:

```
->ASSUME PROCESS $ZNFS;ABORT LAN LAN0
```

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 onto a second line by terminating the first line with an ampersand (&). SCF prompts for additional input before executing the command. For example, the following ADD command takes up three lines:

```
->ADD SERVER $ZNFS.ROOT, STYPE OSS, CPU 0, &
->PRI 148, PROCESS $ROOT, PROGRAM $SYSTEM.ZOSSNFS.NFSSVRHP, &
->MNTPOINT "/"
```

You cannot enter more than 2048 characters for any command.

Note that SCF accepts input from either a terminal or a disk file (command file), and directs output to either a terminal, disk file, or printer. However, for the purposes of this manual, all examples are shown as they appear when a terminal is being used for both input and output. For further information on these other input and output sources, refer to the *Subsystem Control Facility (SCF) Reference Manual* (for D-series releases) or the *SCF Reference Manual for G-Series Releases*.

The EXIT command terminates the SCF session.

Command Descriptions

The remainder of this section describes, in alphabetic order by their command-object pairings, the SCF commands supported by OSS NFS. Each command description includes the command's syntax, the format of the object specifier, and considerations for using the command. For commands that have attribute specifiers, a description of the attribute specifiers is given. Examples are provided for each command.

See Notation Conventions on page xiii for a description of the notation scheme used.

Sensitive and Nonsensitive Commands

Commands can be divided into two categories: sensitive and nonsensitive. Because sensitive commands can change the state of the specified object, sensitive commands can be issued only by the super ID. The following OSS NFS SCF commands are sensitive and require super-ID access:

ADD EXPORT
ADD NETGROUP
ADD USER
ALTER EXPORT
ALTER NETGROUP
ALTER USER
DELETE EXPORT
DELETE NETGROUP
DELETE USER

The following OSS NFS SCF commands are sensitive and require super group access, except as specified above:

ABORT
ADD
ALLOWOPENS
ALTER
DELETE
PRIMARY
START
STATS, with RESET option
STOP
STOPOPENS
TRACE

Nonsensitive commands supported by OSS NFS display information and can be issued by any valid user on the system. The following OSS NFS SCF commands are nonsensitive:

INFO
LISTOPENS
NAMES
STATS, without RESET option
STATUS
VERSION

LIKE, SUB, and SEL Options

The LIKE option is not supported by OSS NFS for any SCF commands. The SUB or SEL option is supported in a command only if shown in that command's syntax description in this section.

The SUB option allows you to restrict the operation of a command to specified subordinate objects (subtypes). You can specify SUB ALL, SUB NONE, SUB ONLY, or SUB `subtype`, where `subtype` is a subordinate object type.

ALL	specifies that the command should be applied to the named object and all subordinate objects.
NONE	specifies that the command should be applied only to the named object.
ONLY	specifies that the command should be applied only to the subordinate objects.
subtype	specifies that the command should be applied only to objects of the specified object type.

If SUB is specified without any option added, ALL is assumed. For specific information on which SUB options are supported or appropriate to each of the commands, see the individual command descriptions.

The SEL option allows you to restrict the operation of a command based on the summary states of objects. If the NOT keyword is included the command is restricted to all objects NOT in the specified summary state. For specific information on which summary states are supported or appropriate to each of the commands, see the individual command descriptions.

Time-Value Format

Time values displayed by commands such as the INFO and STATS commands are presented in the format *HH*:*MM*:*SS*.*hhh*, where *HH* is hours, *MM* is minutes, *SS* is seconds, and *hhh* is thousandths of a second. For example, 11:43:53.775 is 11 hours, 43 minutes, 53 seconds, and 775 thousandths of a second. These time values reflect the local civil time of the system node at which SCF is running.

ABORT LAN Command

The ABORT LAN command terminates the operation of the specified LAN interface process as quickly as possible, regardless of whether network requests are pending or in progress. Only enough processing to ensure the integrity of the subsystem is done. The LAN object is left in the STOPPED summary state.

The ABORT LAN command has the following syntax:

```
ABORT LAN lan-name [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object whose operation is to be terminated.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to abort only LAN objects in the STARTED summary state.

Considerations

Consider the following points when using the ABORT LAN command:

- ABORT LAN is a sensitive command requiring super-group access.
- When you abort a LAN object, it must be in the STARTED summary state.
- This command proceeds even though there are requests pending for a LAN interface process. The consistency of OSS NFS data structures is preserved, but client applications might be disrupted.
- Use the STOP LAN command if you want to stop the operation of LAN objects in a more controlled manner. The STOP LAN command does not abruptly terminate in-progress activities.
- The ABORT LAN command clears the RESTART attribute. The RESTART attribute determines whether or not a subsequent START SUBSYS command initiates an attempt to restart the stopped LAN object. The ABORT LAN command should only be used when you do not want to restart the same configuration of LAN interface and server processes that are currently running. To abort a LAN object without clearing the RESTART attribute, use the ABORT SUBSYS command.
- While a LAN object is stopped, no corresponding LAN interface process exists, and therefore no NFS operations can be accepted over the TCP/IP port serviced by that process.
- Aborting a LAN object does not cause any SERVER objects to be stopped. However, when no LAN object is in the STARTED summary state, SERVER objects in the STARTED summary state are unable to receive or respond to RPC calls.
- Use the START LAN command to reinitiate the operation of the aborted LAN object.
- Use the STATUS LAN command to determine the current summary state of LAN objects.
- Use the DELETE LAN command to remove a LAN object from the OSS NFS subsystem.

Example

The following command aborts a LAN object named $ZNFS.LAN0:

```
->ABORT LAN $ZNFS.LAN0
```

ABORT SERVER Command

The ABORT SERVER command terminates the operation of the specified server processes as quickly as possible, regardless of whether files are open or requests are pending. Only enough processing to ensure the integrity of the subsystem is done. The SERVER objects are left in the STOPPED summary state.

The ABORT SERVER command has the following syntax:

```
ABORT SERVER server-name [ , SEL [ NOT ] state ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object whose operation is to be terminated.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to abort only SERVER objects in the STARTED summary state.

Considerations

Consider the following points when using the ABORT SERVER command:

- When you abort a SERVER object, it must be in the STARTED summary state.
- This command proceeds even though files are open or requests are in progress for the specified server. The consistency of OSS NFS data structures is preserved, but client applications might be disrupted.

  Use the STOP SERVER command if you want to stop the operation of SERVER objects in a more controlled manner. The STOP SERVER command does not abruptly terminate in-progress activities.
- The ABORT SERVER command clears the RESTART attribute. The RESTART attribute determines whether or not a subsequent START SUBSYS command initiates an attempt to restart the stopped SERVER object. The ABORT SERVER command should only be used when you do not want to restart the same configuration of LAN interface and server processes that are currently running. To abort a SERVER object without clearing the RESTART attribute, use the ABORT SUBSYS command.
- While a SERVER object is stopped, no corresponding server process exists. Therefore no NFS operations that deal with the directories and files serviced by that process can be accepted.
- Use the START SERVER command to reinitiate the operation of the aborted server.
- Use the STATUS SERVER command to determine the current summary state of the SERVER objects.
- Use the DELETE SERVER command to remove a SERVER object from the OSS NFS subsystem.

Example

The following command aborts all servers under the assumed manager process that are in the STARTED summary state:

```
->ABORT SERVER $ZNFS.*, SEL STARTED
```

ABORT SUBSYS Command

The ABORT SUBSYS command terminates the operation of the OSS NFS subsystem as quickly as possible, regardless of whether client operations are in progress or remote mounts against the subsystem exist. Only enough processing to ensure the integrity of the subsystem is done. All OSS NFS objects are left in the STOPPED summary state.

The ABORT SUBSYS command has the following syntax:

```
ABORT SUBSYS process-name

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

      [ , SEL [ NOT ] state ]
```

`SUBSYS` *`process-name`*

specifies the name of the SUBSYS object whose operation is to be terminated. The SUBSYS object uses the name of the manager process with which it is associated.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can use the SUB option to specify that you want to abort only specified subordinate objects. ABORT SUBSYS, SUB ALL is equivalent to ABORT SUBSYS, because no further commands can be processed after the subordinate PROCESS object is stopped. With ABORT SUBSYS, only the SUB NONE, SUB ONLY, SUB LAN, and SUB SERVER options are useful.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to abort only objects in the STARTED summary state.

Considerations

Consider the following points when using the ABORT SUBSYS command:

- ABORT SUBSYS is a sensitive command requiring super-group access.
- This command proceeds even though there are remote mounts outstanding against the subsystem, requests pending for a LAN interface process, or requests in progress for servers. The consistency of OSS NFS data structures is preserved, but client applications might be disrupted.

Use the STOP SUBSYS command if you want to stop the operation of OSS NFS objects in a more controlled manner. The STOP SUBSYS command does not abruptly terminate in-progress activities.

- The ABORT SUBSYS command does not clear the RESTART attribute. The RESTART attribute determines whether or not a subsequent START SUBSYS command initiates an attempt to restart the stopped LAN and SERVER objects. ABORT SUBSYS is useful when you want to restart the same configuration of LAN interface and server processes that was running when the command was issued.
- To reinitiate the operation of an aborted OSS NFS subsystem, the manager process must be restarted using a RUN command. After the manager process is restarted, the subsystem can be restored to the same state state it was in prior to the ABORT SUBSYS command by using a START SUBSYS command.
- Use the STATUS SUBSYS command to determine the current summary state of the OSS NFS objects.
- Use the DELETE command on each object to be deleted to remove an object from the OSS NFS subsystem.

Example

The following command aborts the OSS NFS subsystem associated with the manager process $ZNFS:

```
->ABORT SUBSYS $ZNFS
```

ADD EXPORT Command

The ADD EXPORT command adds an EXPORT object to the OSS NFS subsystem.

An EXPORT object allows a file system to be exported to specified netgroups and host machines. Specify *`export-name`* as the OSS NFS pathname of the root directory of the file system to be exported.

The directory is exported when the EXPORT object is created. When a remote mount request is received, a check is made to see if an EXPORT object exists for the mount point of the server responsible for the directory being remotely mounted. If so, and if the EXPORT object permits access to the remote client, then the remote mount is permitted. Otherwise, the client’s mount request is rejected.

If you do not specify an ACCESS attribute when adding an EXPORT object, any host can perform a remote mount of the file system.

The ADD EXPORT command has the following syntax:

```
ADD EXPORT export-name [ [ , attribute-spec ]... ]
```

For an EXPORT object, `attribute-spec` can be:

```
[ ACCESS { name                        }
         { ( name [ , name ]... ) } ]
```

EXPORT *export-name*

specifies the name of the EXPORT object to be added. EXPORT objects are identified by a case-sensitive NFS path name that can include up to 1024 characters, excluding ASCII NUL.

```
ACCESS { name                     }
       { ( name [ , name ]... ) }
```

specifies a netgroup or host machine that is allowed to issue remote mounts against the specified EXPORT object.

Considerations

Consider the following points when using the ADD EXPORT command:

- ADD EXPORT is a sensitive command requiring super-ID access.
- You cannot use object-name templates to specify object names with the ADD EXPORT command.
- The EXPORT object is only meaningful when its object name corresponds to a directory that is the mount point of a SERVER object in the STARTED summary state. While you can define EXPORT objects for non-existent directories or for directories that are not server mount points, these EXPORT objects are ignored. Export permissions are only checked by the NFS manager process for the mount point of the server responsible for the directory being remotely mounted.

Examples

This command allows a host machine named sysabc to perform a remote mount of the root directory:

```
->ADD EXPORT $ZNFS./, ACCESS sysabc
```

This command allows the netgroups named sales and dist to perform a remote mount of the /usr/invoice directory:

```
->ADD EXPORT $ZNFS./usr/invoice, ACCESS (sales, dist)
```

ADD GROUP Command

The ADD GROUP command adds a GROUP object to the OSS NFS subsystem.

GROUP objects are used to classify groups of NFS users to define NETGROUP objects and to translate group IDs to group names. A group has a name and a numeric ID and is composed of one or more users with unique names.

The ADD GROUP command has the following syntax:

```
ADD GROUP group-name [ [ , attribute-spec ]... ]
```

For a GROUP object, *attribute-spec* can be:

```
GROUPID group-id
[ MEMBER { user-name                         }
         { ( user-name [ , user-name ]... ) } ]
```

`GROUP` *`group-name`*

specifies the name of the GROUP object to be added. GROUP objects are identified by a case-sensitive NFS group name containing 1 to 64 ASCII characters, excluding ASCII NUL and colon(:). Group names must begin with a letter, numeral, or underscore (_).

`GROUPID` *`group-id`*

specifies the number of the group. A group ID can be from 0 through 4294967295. No two groups can have the same group number.

```
MEMBER { user-name                         }
       { ( user-name [ , user-name ]... ) }
```

specifies the names of users that are members of the group. The ADD GROUP command does not create users; you create users by using the ADD USER command. You can also add members later by using the ALTER GROUP command.

Considerations

Consider the following points when using the ADD GROUP command:

- ADD GROUP is a sensitive command requiring super-group access.
- You cannot use object-name templates to specify object names with the ADD GROUP command.

Examples

The following command adds a group named sales with three members and group ID 25:

```
->ADD GROUP $ZNFS.sales, MEMBER (susie, phil, alan)&
->, GROUPID 25
```

The following command adds a group named software with group ID 50:

```
->ADD GROUP $ZNFS.software, GROUPID 50
```

The members are added later by using the ALTER GROUP command.

ADD LAN Command

The ADD LAN command adds a LAN object to the OSS NFS subsystem.

A LAN object specifies a local area network (LAN) interface process that acts as a front end for NFS requests received from the NonStop TCP/IP communications process.

After you have added a LAN object, it will be in the STOPPED summary state. Use the START LAN command to start the LAN object.

The ADD LAN command has the following syntax:

```
ADD LAN lan-name [ [ , attribute-spec ]... ]
```

For a LAN object, *attribute-spec* can be any of:

```
CPU cpu-number
PROCESS process-name
PROGRAM file-name
[ ADDR-CHECK { ON | OFF } ]
[ BACKUP backup-cpu-number ]
[ DOMAIN { domain-name                            }
         { ( domain-name [ , domain-name ]... ) } ]
[ HISTOGRAM ( message-length [ , message-length ] ... ) ]
[ PRI cpu-priority ]
[ SWAP file-name ]
[ TCPIP-HOST-FILE tcpip-host-file ]
[ TCPIP-PROCESS-NAME tcpip-process-name ]
[ TCPIP-RESOLVER-NAME tcpip-resolver-name ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object to be added. LAN objects are identified by a case-insensitive logical name starting with an alphabetic character and containing one to eight alphanumeric characters.

`CPU` *`cpu-number`*

specifies the number of the primary processor on which the LAN interface process runs. You can specify from 0 through 15.

`PROCESS` *`process-name`*

specifies the name of the LAN interface process.

`PROGRAM` *`file-name`*

specifies the NonStop Kernel file name of the LAN interface program. Use the file name that was used to install the NFSLAN file. See the *Open System Services NFS Management and Operations Guide* for more information about installation.

`ADDR-CHECK { ON | OFF }`

specifies whether the host name is checked when a mount or unmount request is received. If ON is specified, a check is made to determine whether the host name corresponds with the Internet Protocol (IP) address of the request. When set to ON, a helper process is started and stopped each time the LAN object is started and stopped. The default value is ON.

`BACKUP` *`backup-cpu-number`*

specifies the number of the backup processor on which the LAN interface process should be restarted if the primary processor fails. You can specify from -1 through 15. The backup processor must be a different processor than the primary processor. A value of -1 indicates that no backup processor is designated. If you do not specify a backup processor, the LAN interface process does not run as a restartable process.

```
DOMAIN { domain-name                             }
       { ( domain-name [ , domain-name ]... ) }
```

specifies one or more domains. These domain names correspond to the domain names specified when defining NETGROUP objects. See the ADD NETGROUP command for more information.

`HISTOGRAM (` *`message-length`* `[ ,` *`message-length`* `]... )`

specifies up to six values that measure NFS message lengths (in bytes). The values specified are used to define the bounds of up to seven buffers. Each buffer is used to produce the histogram statistics displayed by the STATS LAN command. You can specify the values in any order; duplicate values are ignored.

The values are arranged in ascending order and used to define the bounds of a corresponding set of buffers. The number of buffers is one greater than the number of *`message-length`* values you specify. For example, if you specify HISTOGRAM 500, two buffers are used—one for messages less than or equal to 500 bytes in length and one for messages greater than 500 bytes in length. You can have up to seven buffers.

If you omit this attribute, the default values are 128, 256, 512, 1024, 2048, and 4096 bytes. Seven buffers are used. (See the STATS LAN command for an example.)

`PRI` *`cpu-priority`*

specifies the execution priority for the LAN interface process. You can specify a priority of -1 or a priority in the range 1 through 199. (Processes with higher numbers have higher priority.) If you specify a priority of -1 or omit this attribute, the priority of the NFS manager process at the time the LAN object is started is used as the priority of the LAN interface process.

`SWAP` *`file-name`*

specifies a file used to hold the LAN interface process's virtual data. This attribute allows you to specify a permanent file for swapping the data stack and to specify a different volume for the swap file. If you omit this attribute, the volume on which the program file is located is used.

See the RUN command in the *TACL Reference Manual* for more information about specifying a swap file.

`TCPIP-HOST-FILE` *`tcpip-host-file`*

specifies the name of the TCP/IP HOSTS file. If you omit this attribute, NonStop TCP/IP uses a domain name resolver.

`TCPIP-PROCESS-NAME` *`tcpip-process-name`*

specifies the name of the TCP/IP process (for conventional NonStop TCP/IP) or the TCPSAM process (for Parallel Library TCP/IP) to be used by the LAN interface process. If you omit this attribute, the $ZTC0 process is used. See the *TCP/IP Configuration and Management Manual* for information about conventional NonStop TCP/IP. See the *TCP/IP (Parallel Library) Configuration and Management Manual* for information about Parallel Library TCP/IP.

`TCPIP-RESOLVER-NAME` *`tcpip-resolver-name`*

specifies the name of the TCP/IP resolver configuration file. If you omit this attribute, the name $SYSTEM.ZTCPIP.RESCONF is used. For descriptions of the TCP/IP files, see the *Tandem TCP/IP Configuration and Management Manual* (for conventional NonStop TCP/IP) or the *TCP/IP (Parallel Library) Configuration and Management Manual*.

Considerations

Consider the following points when using the ADD LAN command:

- ADD LAN is a sensitive command requiring super-group access.
- You cannot use object-name templates to specify object names with the ADD LAN command.
- For conventional NonStop TCP/IP, you can define more than one LAN object, but you can start only one LAN interface process for each TCP/IP process at any one time. For Parallel Library TCP/IP, you can define more than one LAN object, but you cannot define more than one LAN interface process for the entire Parallel Library TCP/IP subsystem.

Examples

The following command adds a LAN object named LAN0 that runs on primary processor 2 with an execution priority of 170. If the primary processor fails and the LAN interface process is restarted, the LAN interface process will run on processor 3. The LAN interface process name is $LAN0 and the program name is $SYSTEM.ZOSSNFS.NFSLAN. The histogram statistics are collected in three buffers—messages less than or equal to 512 bytes, messages less than or equal to 1024 bytes, and messages greater than 1024 bytes:

```
->ADD LAN $ZNFS.LAN0, CPU 2, BACKUP 3 &
->, PRI 170, PROCESS $LAN0 &
->, PROGRAM $SYSTEM.ZOSSNFS.NFSLAN &
->, HISTOGRAM (512, 1024)
```

The next command adds a LAN object named NFSLAN that runs on primary processor 1 with priority 170 and program name $SYSTEM.ZOSSNFS.NFSLAN. The LAN interface process runs without a restartable process:

```
->ADD LAN $ZNFS.NFSLAN, CPU 1, PRI 170, PROCESS $LAN0 &
->, PROGRAM $SYSTEM.ZOSSNFS.NFSLAN
```

ADD NETGROUP Command

The ADD NETGROUP command adds a NETGROUP object to the OSS NFS subsystem.

A NETGROUP object defines a group of host systems within specified network domains. NETGROUP objects can be used to specify the access list associated with EXPORT objects.

The ADD NETGROUP command has the following syntax:

```
ADD NETGROUP netgroup-name [ [ , attribute-spec... ]
```

For a NETGROUP object, *attribute-spec* can be any of:

```
[ netgroupx-name ]
[ ( [ host-name ] , [ user-name ] , [ domain-name ] ) ]
```

NETGROUP *netgroup-name*

specifies the name of the NETGROUP object to be added. NETGROUP objects are identified by a case-insensitive name containing 1 to 64 ASCII characters, excluding ASCII NUL.

netgroupx-name

specifies the name of another netgroup whose members are also members of the netgroup you are defining. You cannot define netgroups recursively.

`( [` *`host-name`* `] , [` *`user-name`* `] , [` *`domain-name`* `] )`

is a list that adds a specific host machine to the netgroup you are defining. The *`user-name`* and *`domain-name`* attributes are accepted; however, they are reserved for future use.

If you specify any name that does not start with a letter, digit, or underscore (_), all objects of that type are excluded from the netgroup.

You can specify multiple *`netgroupx-names`* and lists as attributes of the same NETGROUP object.

Considerations

Consider the following points when using the ADD NETGROUP command:

- ADD NETGROUP is a sensitive command requiring super-ID access.
- You cannot use object-name templates to specify object names with the ADD NETGROUP command.
- Host and domain names are case-sensitive, externally defined names that can contain up to 255 characters. Because these names are defined by the NonStop TCP/IP subsystem, the OSS NFS subsystem does not impose any restrictions on the characters that can be contained in these names; however, other network software might not work with names that contain certain nonalphanumeric characters or names that exceed a certain length.

Example

The following command adds a netgroup named labtechs. The members are all users of the lxn1 machine in the medlab domain:

```
->ADD NETGROUP $ZNFS.labtechs, (lxn1, , medlab)
```

ADD SERVER Command

The ADD SERVER command adds a SERVER object to the OSS NFS subsystem.

A SERVER object defines an NFS server.

After you have added a SERVER object, it will be in the STOPPED summary state. Use the START SERVER command to start the object.

The ADD SERVER command has the following syntax:

```
ADD SERVER server-name [ [ , attribute-spec ]... ]
```

For a SERVER object, *attribute-spec* can be any of:

```
CPU cpu-number
MNTPOINT path-name
PROGRAM file-name
PROCESS process-name
STYPE OSS
[ BACKUP backup-cpu-number ]
[ FILE-OPT-OK { TRUE | FALSE } ]
[ MAX-FILE-SIZE integer ]
[ NULL-ALIAS-OK { TRUE | FALSE } ]
[ PRI cpu-priority ]
[ READ-ONLY { TRUE | FALSE } ]
[ ROOT-USER-OK { TRUE | FALSE } ]
[ SWAP file-name ]
[ UPSHIFT { TRUE | FALSE } ]
[ WRITE-THRU { TRUE | FALSE } ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object to be added. SERVER objects are identified by a case-insensitive logical name starting with an alphabetic character and containing one to eight alphanumeric characters.

`CPU` *`cpu-number`*

specifies the number of the primary processor on which the server process runs. You can specify from 0 through 15.

`MNTPOINT` *`path-name`*

specifies the pathname of the directory file on which the server's file system is to be mounted. Before the server can be started, this directory must exist.

`PROGRAM` *`file-name`*

specifies the NonStop Kernel filename of the server program. Use the filename that was specified when the NFSSVRHP file was installed. See the *Open System Services NFS Management and Operations Guide* for more information about installation.

`PROCESS` *`process-name`*

specifies the name of the server process.

`STYPE OSS`

specifies that the the type of server is an OSS NFS file server. NFS file servers store and provide access to NFS files, which are typically in formats compatible with PCs or workstations: that is, files compatible with DOS or UNIX.

`BACKUP` *`backup-cpu-number`*

specifies the number of the backup processor on which the server process runs. You can specify from -1 through 15. The backup processor must be a different processor than the primary processor. A value of -1 indicates that no backup processor is designated. If you do not specify a backup processor, the server process does not run in persistent mode.

`FILE-OPT-OK { TRUE | FALSE }`

specifies whether a client user can include a list of file options at the end of a file name. OSS NFS uses the options to determine characteristics and the access type for the file. A client user specifies the options as a list of keywords (separated by commas) following a file name. (See the *Overview of NFS for Open System Services*) for a complete description of the format for entering file options.) You can define server-default values for these file options by specifying the ACCESSTYPE, FILECODE, PRIMEXT, SEC-EXT, and MAXEXTENTS attributes.

The default value is FALSE—file options cannot be included.

`MAX-FILE-SIZE` *`integer`*

specifies the maximum size, in bytes, of files written by NFS. You can specify any valid 32-bit integer. The default value is 0 indicating no limit.

`NULL-ALIAS-OK { TRUE | FALSE }`

specifies whether remote NFS users who do not have a mapped user ID are allowed access to files managed by the server. If TRUE, these users are mapped to the "nobody" user ID (-2), which must be defined in the list of OSS NFS user IDs. If FALSE, a user must have a mapped user ID to gain access. See the ADD USER command for information on mapped user IDs. The default value is FALSE.

`PRI` *`cpu-priority`*

specifies the execution priority for the server process. You can specify a priority of -1 or a priority from 1 through 199. (Processes with higher numbers have higher priority.) If you specify a priority of -1 or omit this attribute, the priority of the NFS manager process at the time the SERVER object is started is used as the priority of the server process.

`READ-ONLY { TRUE | FALSE }`

specifies whether or not access to the file system serviced by the server is on a read-only or read/write basis. If TRUE, the file system must be initialized before you create this server by a server that is not configured as read-only. The default value is FALSE—the file system allows read/write access.

`ROOT-USER-OK { TRUE | FALSE }`

specifies whether remote root users are allowed to access the server’s files with super-ID privileges. Root users have access to all NFS files, regardless of their permission and ownership values. This super-ID permission is normally not allowed during access checking because anyone who can become the root user on their workstation can gain access to any remote file. If TRUE, remote root users are allowed access to the server's files with super-ID privileges. If FALSE, requests from remote root users are treated as requests from the NFS nobody user (user ID 65534). Note that the FALSE value is contingent on the definition of user ID 65534 as the nobody user ID. To prevent super-ID permission during access checking, a nobody user ID must be defined.

The default value is FALSE.

`SWAP` *`file-name`*

specifies a file used to hold the server process's virtual data. This attribute allows you to specify a permanent file for swapping the data stack and to specify a different volume for the swap file. See the RUN command in the *TACL Reference Manual* for more information about specifying a swap file.

`UPSHIFT { TRUE | FALSE }`

specifies whether file names on the NonStop Himalaya system are upshifted when presented to NFS clients.

The default value is FALSE.

`WRITE-THRU { TRUE | FALSE }`

specifies whether data written by NFS clients to NonStop Himalaya systems is written to cache or directly to disk. Configurations involving high-performance applications should set this parameter to TRUE while those involving high-reliability applications should set this parameter to false.

The default value is FALSE.

Considerations

Consider the following points when using the ADD SERVER command:

- ADD SERVER is a sensitive command requiring super-group access.
- You cannot use object-name templates to specify object names with the ADD SERVER command.
- NFS pass-through server processes always refer to the ZNFSUSR and ZNFSUSR1 files that are on the system they are running on. If the configuration includes pass-through servers that are on a different *Expand* node than the one that the manager process is on, the ZNFSUSR and ZNFSUSR1 files from the manager node must be copied to the server node and properly secured. Each time any USER objects are changed, the files must be copied over again for the remote Expand server to receive the changes.

Table 2-2 describes the interactions between the "nobody" alias and the NULL-ALIAS-OK parameter.

Table 2-2. NULL-ALIAS-OK Interactions With the "Nobody" Alias

	NULL-ALIAS-OK = TRUE	NULL-ALIAS-OK = FALSE
"nobody" alias defined	The "nobody" NFS user ID is mapped to the "nobody" alias user ID	NFS returns status condition NFSERR_PERM to the client indicating a permission violation.
"nobody" alias not defined	The "nobody" NFS user ID is not mapped.	NFS returns status condition NFSERR_PERM to the client indicating a permission violation.

Example

The following command adds an NFS server named NSRVR3. The server process name is $NSRV3, and the program name is $SYSTEM.ZOSSNFS.NFSSVRHP. The primary processor is 0, and the backup processor is 1. The server uses the local OSS file set under `/usr`:

```
->ADD SERVER $ZNFS.NUSR, STYPE OSS, CPU 0, BACKUP 1 &
-> , PRI 148, PROCESS $NUSR &
-> , PROGRAM $SYSTEM.ZOSSNFS.NFSSVRHP &
-> , MNTPOINT /usr
```

ADD USER Command

The ADD USER command adds a USER object to the OSS NFS subsystem.

A USER object registers a user with the OSS NFS subsystem and maps the user's name to an NFS user ID.

The ADD USER command has the following syntax:

```
ADD USER user-name [ [ , attribute-spec ]... ]
```

For a USER object, *attribute-spec* can be any of:

```
GROUPID group-id
USERID user-id
[ ALIAS OSS { user-name   }
            { user-number } ]
[ COMMENT "string" ]
```

`USER` *`user-name`*

specifies the name of the USER object to be added. USER objects are identified by a case-sensitive logical name starting with an alphabetic character, a number, or an underscore (_), and containing 1 to 64 alphanumeric characters, excluding ASCII NUL and colon (:).

`GROUPID` *`group-id`*

specifies the user's group number, which can be a value from 0 through 4294967295. An NFS user can, but need not, be a member of a group; that is, although this attribute must be specified, a corresponding predefined group is not required.

`USERID` *`user-id`*

specifies the user number, which can be a value from 0 through 4294967295. Two values are recognized as special by the OSS NFS subsystem: 0 is for the root or super-ID user, and 65534 is for the nobody user. Note that some clients use the value 4294967294 (the unsigned equivalent of -2) instead of 65534.

```
ALIAS OSS { user-name   }
          { user-number }
```

specifies a NonStop Kernel user ID, referred to as the mapped user ID, to be used on behalf of the NFS user when requesting file access. The NonStop Kernel user ID must already be defined in the USERID file on the NonStop Himalaya system on which the OSS NFS manager process is running. *`user-name`* must be either a NonStop Kernel user name or a NonStop Kernel user alias; *`user-number`* must be the scalar form of a NonStop Kernel user ID.

The scalar form of a NonStop Kernel user ID is obtained by using the following formula:

```
( group-number x 256) + member-number.
```

For example, the scalar form of the NonStop Kernel user ID 10,10 is:

```
(10 x 256) + 10 = 2570
```

If you omit the ALIAS OSS attribute, the user has no mapped user ID.

`COMMENT` *`"string"`*

is an optional attribute that allows you to include information about the user. The information is in readable ASCII format and can be from 1 through 128 characters long. For example, you might use this attribute to record the user's full name and office extension.

Considerations

Consider the following points when using the ADD USER command:

- ADD USER is a sensitive command requiring super-ID access.
- You cannot use object-name templates to specify object names with the ADD USER command.
- To add a USER object, you must be the super ID.

Examples

The following command adds a user named susie whose user ID is 118 and whose group ID is 25:

```
->ADD USER $ZNFS.susie, USERID 118, GROUPID 25 &
-> , COMMENT “Susie Shaw”
```

The next command adds a user named phil whose user ID is 35, whose group ID is 25, and whose mapped user name is SALES.PHIL:

```
->ADD USER $ZNFS.phil, USERID 35, GROUPID 25 &
-> , COMMENT “Phil Chomsky”, ALIAS OSS SALES.PHIL
```

ALLOWOPENS SUBSYS Command

The ALLOWOPENS SUBSYS command allows opens to be issued to the specified OSS NFS subsystem; that is, the subsystem can accept new remote mount requests from clients. This command reverses the effect of the STOPOPENS SUBSYS command.

The ALLOWOPENS SUBSYS command has the following syntax:

```
ALLOWOPENS [ SUBSYS ] process-name
```

`[ SUBSYS ]` *`process-name`*

specifies the name of the SUBSYS object which is to allow new remote mounts. The SUBSYS object uses the name of the manager process with which it is associated.

Considerations

Consider the following points when using the ALLOWOPENS SUBSYS command:

- ALLOWOPENS SUBSYS is a sensitive command requiring super-group access.
- No particular summary state is required for this command.

Example

The following command allows the OSS NFS subsystem associated with the manager process $ZNFS to accept new remote mount requests:

```
->ALLOWOPENS SUBSYS $ZNFS
```

ALTER EXPORT Command

The ALTER EXPORT command changes the remote client access of the specified EXPORT object.

The ALTER EXPORT command has the following syntax:

```
ALTER EXPORT export-name [ [ , attribute-spec] ... ]
```

For an EXPORT object, *attribute-spec* can be any of:

```
[ ADD-ACCESS [ name                      ]
             [ ( name [ , name ] ... ) ] ]
[ DEL-ACCESS [ name                      ]
              [ ( name [ , name ] ... ) ] ]
```

```
EXPORT export-name
```

specifies the name of the EXPORT object whose access list is to be altered.

```
ADD-ACCESS [ name                      ]
            [ ( name [ , name ] ... ) ]
```

specifies one or more netgroup or host machine names to be added to the access list of the EXPORT object.

```
DEL-ACCESS [ name                      ]
            [ ( name [ , name ] ... ) ]
```

specifies one or more netgroup or host machine names to be deleted from the access list of the EXPORT object.

Considerations

Consider the following points when using the ALTER EXPORT command:

- ALTER EXPORT is a sensitive command requiring super-ID access.
- The ALTER EXPORT command processes all ADD-ACCESS specifications before all DEL-ACCESS specifications. Duplicate ADD-ACCESS or DEL-ACCESS specifications cause a warning to be returned.
- Use the INFO EXPORT command to determine the current remote client access of an EXPORT object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.
- If all members of an EXPORT access list are deleted, the EXPORT object is exported to everyone. If the intent is to disallow all access to the EXPORT object,

the EXPORT object itself should be deleted by using the DELETE EXPORT command.

Example

The following command alters the EXPORT access list for the server mounted at /usr as follows:

- Allows a netgroup named labs and a host machine named warehs to issue remote mounts against directories maintained by that server.
- Disallows remote mounts of that server by the host machine named dist.

```
->ALTER EXPORT $ZNFS./usr, ADD-ACCESS (labs, warehs) &
-> , DEL-ACCESS dist
```

ALTER GROUP Command

The ALTER GROUP command changes attribute values associated with the specified GROUP object.

The ALTER GROUP command has the following syntax:

```
ALTER GROUP group-name [ [ , attribute-spec ]... ]
```

For a GROUP object, *attribute-spec* can be:

```
[ ADD-MEMBER { user-name                            }
             { ( user-name [ , user-name ]... ) } ]
[ DEL-MEMBER { user-name                            }
             { ( user-name [ , user-name ]...  )  } ]
```

GROUP *group-name*

specifies the name of the GROUP object whose attributes are to be altered.

```
ADD-MEMBER { user-name                            }
           { ( user-name [ , user-name ]... ) }
```

specifies the names of group members to be added to the group.

```
DEL-MEMBER { user-name                            }
           { ( user-name [ , user-name ]... ) }
```

specifies the names of group members to be deleted from the group.

Considerations

Consider the following points when using the ALTER GROUP command:

- ALTER GROUP is a sensitive command requiring super-group access.

- The ALTER GROUP command processes all ADD-MEMBER specifications before all DEL-MEMBER specifications. Duplicate ADD-MEMBER or DEL-MEMBER specifications cause a warning to be returned.
- Use the INFO GROUP command to determine the current attribute values of a GROUP object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.

Example

The following command adds members named garyt and marilyn to the software group and deletes a member named andy:

```
->ALTER GROUP $ZNFS.software, ADD-MEMBER ( garyt &
->,marilyn ), DEL-MEMBER andy
```

ALTER LAN Command

The ALTER LAN command changes attribute values associated with the specified LAN object.

The ALTER LAN command has the following syntax:

```
ALTER LAN lan-name [ [ , attribute-spec ]... ]

  [ , SEL [ NOT ] state ]
```

For a LAN object, *attribute-spec* can be any of:

```
[ ADDR-CHECK { ON | OFF } ]
[ ADD-DOMAIN { domain-name                           }
             { ( domain-name [ , domain-name ]... ) } ]
[ BACKUP backup-cpu-number ]
[ CPU cpu-number ]
[ DEL-DOMAIN { domain-name                           }
             { ( domain-name [ , domain-name ]... ) }  ]
[ HISTOGRAM ( message-length [ , message-length ]... ) ]
[ PRI cpu-priority ]
[ PROCESS process-name ]
[ PROGRAM file-name ]
[ SWAP [ file-name ] ]
[ TCPIP-HOST-FILE [ tcpip-host-file ] ]
[ TCPIP-PROCESS-NAME [ tcpip-process-name ] ]
[ TCPIP-RESOLVER-NAME [ tcpip-resolver-name ] ]

      [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object whose attributes are to be altered.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to alter only LAN objects in the STOPPED summary state.

`ADDR-CHECK { ON | OFF }`

specifies whether the host name is checked when a mount or unmount request is received. If ON is specified, a check is made to determine whether the host name corresponds with the Internet Protocol (IP) address of the request. When set to ON, a helper process is started and stopped each time the LAN object is started and stopped. The default value is ON.

```
ADD-DOMAIN { domain-name                            }
           { ( domain-name [ , domain-name ]... ) }
```

specifies one or more domains to be added to the list of domains. See ALTER NETGROUP Command on page 2-28 for more information on the use of domains.

`BACKUP` *`backup-cpu-number`*

specifies the number of the backup processor on which the LAN interface process should be restarted if the primary processor fails. You can specify from -1 through 15. The backup processor must be a different processor than the primary processor. A value of -1 indicates that no backup processor is designated. If you do not specify a backup processor, the LAN interface process is not restarted in case of failure.

`CCPU` *`cpu-number`*

specifies the number of the primary processor on which the LAN interface process runs. You can specify from 0 through 15.

```
DEL-DOMAIN { domain-name                            }
           { ( domain-name [ , domain-name ]... ) }
```

specifies one or more domains to be deleted from the list of domains.

`HISTOGRAM (` *`message-length`* `[ ,` *`message-length`* `]... )`

specifies up to six values that measure NFS message lengths (in bytes). The values specified are used to define the bounds of up to seven buffers. Each buffer is used to produce the histogram statistics displayed by the STATS LAN command. You can specify the values in any order; duplicate values are ignored.

The values are arranged in ascending order and used to define the bounds of a corresponding set of buffers. The number of buffers is one greater than the number of *`message-length`* values you specify. For example, if you specify HISTOGRAM 500, two buffers are used—one for messages less than or equal to 500 bytes in length and one for messages greater than 500 bytes in length. You can have up to seven buffers.

If you omit this attribute, the default values are 128, 256, 512, 1024, 2048, and 4096 bytes. Seven buffers are used. (See STATS LAN Command on page 2-71 for an example.)

`PRI` *`cpu-priority`*

specifies the execution priority for the LAN interface process. You can specify a priority of -1 or a priority in the range 1 through 199. (Processes with higher numbers have higher priority.) If you specify a priority of -1 or omit this attribute, the priority of the NFS manager process at the time the LAN object is started is used as the priority of the LAN interface process.

`PROCESS` *`process-name`*

specifies the name of the LAN interface process.

`PROGRAM` *`file-name`*

specifies the NonStop Kernel file name of the LAN interface program. Use the file name that was used to install the NFSLAN file. See the *Open System Services NFS Management and Operations Guide* for more information about installation.

`SWAP [`*`file-name`*`]`

specifies a file used to hold the LAN interface process's virtual data. This attribute allows you to specify a permanent file for swapping the data stack and to specify a different volume for the swap file. If you omit this attribute, the volume on which the program file is located is used.

See the RUN command in the *TACL Reference Manual* for more information about specifying a swap file.

`TCPIP-HOST-FILE [`*`tcpip-host-file`*`]`

specifies the name of the TCP/IP HOSTS file. If you omit this attribute, NonStop TCP/IP uses a domain name resolver.

`TCPIP-PROCESS-NAME [`*`tcpip-process-name`*`]`

specifies the name of the TCP/IP process (for conventional NonStop TCP/IP) or the TCPSAM process (for Parallel Library TCP/IP) to be used by the LAN interface process. If you omit this attribute, the $ZTC0 process is used. See the *TCP/IP Configuration and Management Manual* for information about conventional NonStop TCP/IP. See the *TCP/IP (Parallel Library) Configuration and Management Manual* for information about Parallel Library TCP/IP.

`TCPIP-RESOLVER-NAME [`*`tcpip-resolver-name`*`]`

specifies the name of the TCP/IP resolver configuration file. If you omit this attribute, the name $SYSTEM.ZTCPIP.RESCONF is used. For descriptions of the TCP/IP files, see the *TCP/IP Configuration and Management Manual* (for conventional NonStop TCP/IP) or the *TCP/IP (Parallel Library) Configuration and Management Manual*.

Considerations

Consider the following points when using the ALTER LAN command:

- ALTER LAN is a sensitive command requiring super-group access.
- When you alter a LAN object, it must be in the STOPPED summary state.
- If you specify any of the SWAP, TCPIP-PROCESS-NAME, TCPIP-RESOLVER-NAME, and TCPIP-HOST-FILE attributes without an argument, any existing specification for that attribute is deleted.
- Use the INFO LAN command to determine the current attribute values of a LAN object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.
- To move the backup LAN interface process from one processor to another, two ALTER LAN commands are required. The current backup process must first be stopped by issuing an ALTER LAN command that sets the BACKUP attribute to -1. The backup process can then be restarted in the target process by issuing an ALTER LAN command that sets the BACKUP attribute to the value of the target processor.

Example

The following command alters the LAN object named LAN0, setting the priority to 140 and the LAN program name to $ALTVOL.SYSTEM.NFSLAN:

```
->ALTER LAN $ZNFS.LAN0, PRI 140 &
-> , PROGRAM $ALTVOL.SYSTEM.NFSLAN
```

ALTER NETGROUP Command

The ALTER NETGROUP command changes the membership of the specified NETGROUP object.

The ALTER NETGROUP command has the following syntax:

```
ALTER NETGROUP netgroup-name [ [ , attribute-spec ]... ]
```

For a NETGROUP object, *attribute-spec* can be any of:

```
[ { ADD | DEL } netgroupx-name ]
[ { ADD | DEL } ( [host-name], [user-name], [domain-name] ) ]
```

NETGROUP *netgroup-name*

specifies the name of the NETGROUP object whose membership is to be altered.

`ADD` *`netgroupx-name`*

adds another netgroup to the specified netgroup. Recursive definitions are not allowed.

`DEL` *`netgroupx-name`*

deletes a netgroup from the specified netgroup.

`ADD ( [` *`host-name`* `] , [` *`user-name`* `] , [` *`domain-name`* `] )`

adds a specific host to the netgroup. The *`user-name`* and *`domain-name`* attributes are accepted; however, they are reserved for future use. See “ADD NETGROUP Command,” earlier in this section, for more information about specifying this list.

`DEL ( [` *`host-name`* `] , [` *`user-name`* `] , [` *`domain-name`* `] )`

deletes a specific host from the netgroup. You must specify the triple (list) exactly as it is defined in the current EXPORT object. You cannot omit an object from the list to specify a wild-card element. For example, deleting the triple (, janet, medlab) does not delete the triple (sysabc, janet, medlab) even though the triple you specified includes (sysabc, janet, medlab).

Considerations

Consider the following points when using the ALTER NETGROUP command:

- ALTER NETGROUP is a sensitive command requiring super-ID access.
- You can specify the NETGROUP attributes multiple times, in any order. The subsystem first processes the ADD attributes and then processes the DEL attributes. Duplicate specifications cause a warning to be returned.
- Use the INFO NETGROUP command to determine the current membership of a NETGROUP object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.

Example

The following command adds to the netgroup named medaccts a user named wes of a host machine named forty in the medlab domain:

```
->ALTER NETGROUP $ZNFS.medaccts &
-> , ADD (forty, wes, medlab)
```

ALTER PROCESS Command

The ALTER PROCESS command changes attribute values associated with the specified PROCESS object.

The ALTER PROCESS command has the following syntax:

```
ALTER PROCESS process-name [ [ , attribute-spec ]... ]

      [ , SEL [ NOT ] state ]
```

For a PROCESS object, *attribute-spec* can be any of:

```
[ BACKUP [ backup-cpu-number ] ]
[ BACKUPDEBUG { ON | OFF } ]
[ COLLECTOR [ process-name ] ]
[ DEBUGONERR { ON | OFF } ]
[ LOGFILE [ file-name ] ]
[ MSGFILE file-name ]
```

`PROCESS` *`process-name`*

specifies the name of the PROCESS object whose attributes are to be altered.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to alter only PROCESS objects in a specified summary state. See Summary States on page 1-7, for the names and descriptions of these states.

`BACKUP [` *`backup-cpu-number`* `]`

specifies the number of the backup processor on which the manager process runs. You can specify from -1 through 15. If you specify -1 or do not specify this attribute, the manager process does not run as a process pair.

`BACKUPDEBUG { ON | OFF }`

specifies that any subsequent backup should be initiated in debug mode.

△ **Caution.** Use BACKUPDEBUG carefully, because turning it on inhibits takeover and prevents normal fault-tolerant operation.

`COLLECTOR [`*`process-name`*`]`

specifies the name of the Event Management Service (EMS) collector process. If the COLLECTOR attribute is specified without an argument, EMS logging is stopped.

`DEBUGONERR { ON | OFF }`

specifies what happens when a fatal error occurs. If ON, the process goes into debug mode. If OFF, the process aborts (the ABEND routine is called).

△ **Caution.** Use DEBUGONERR carefully, because turning it on inhibits takeover and prevents normal fault tolerant operation.

`LOGFILE [`*`file-name`*`]`

specifies the name of the log file used to used by the manager process. If the LOGFILE attribute is specified without an argument, logging to the log file is stopped.

`MSGFILE` *`file-name`*

specifies the name of the message file to be used by the manager process. This file is usually located on the same subvolume as the OSS NFS manager process and its value is not usually altered.

Considerations

Consider the following points when using the ALTER PROCESS command:

- ALTER PROCESS is a sensitive command requiring super-group access.
- The COLLECTOR attribute for the PROCESS object controls the EMS collector for all LAN and SERVER objects as well. All the events generated by the subsystem are sent to the same collector. Any changes to the collector are propagated to all LAN and SERVER objects in the STARTED summary state.
- The LOGFILE attribute is a feature of the manager process and only contains events generated by the manager process. Events generated by a LAN or SERVER object never appear in the log file.
- Use the INFO PROCESS command to determine the current attribute values of the PROCESS object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.

Example

The following command turns off the DEBUGONERROR attribute and specifies a log file:

```
->ALTER PROCESS $ZNFS, DEBUGONERROR OFF, &
->LOGFILE $VOL5.LOGNFS.LOG1
```

ALTER SERVER Command

The ALTER SERVER command changes attribute values associated with the specified SERVER object.

The ALTER SERVER command has the following syntax:

```
ALTER SERVER server-name [ [ , attribute-spec ]... ]

      [ , SEL [ NOT ] state ]
```

For a SERVER object, *attribute-spec* can be:

```
[ CPU cpu-number ]
[ BACKUP backup-cpu-number ]
[ MAX-FILE-SIZE integer ]
[ MNTPOINT path-name ]
[ NULL-ALIAS-OK { TRUE | FALSE } ]
[ PRI cpu-priority ]
[ PROGRAM file-name ]
[ PROCESS process-name ]
[ READ-ONLY { TRUE | FALSE } ]
[ ROOT-USER-OK { TRUE | FALSE } ]
[ STYPE OSS ]
[ SWAP [ file-name ] ]
[ WRITE-THRU { TRUE | FALSE } ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object whose attributes are to be altered.

`CPU` *`cpu-number`*

specifies the number of the primary processor on which the server process runs. You can specify from 0 through 15.

`BACKUP` *`backup-cpu-number`*

specifies the number of the backup processor on which the server process runs. You can specify from -1 through 15. The backup processor must be a different processor than the primary processor. A value of -1 indicates that no backup processor is designated. If you do not specify a backup processor, the server process does not run in persistent mode.

`MAX-FILE-SIZE` *`integer`*

specifies the maximum size, in bytes, of files written by NFS. You can specify any valid 32-bit integer. The default value is 0 indicating no limit.

`MNTPOINT` *`path-name`*

specifies the path name of the directory file on which the server's file system is to be mounted. Before the server can be started, this directory must exist.

`NULL-ALIAS-OK { TRUE | FALSE }`

specifies whether remote NFS users who do not have an alias are allowed access to files managed by the server. If TRUE, these users are mapped to the "nobody" user ID (-2). The "nobody" user ID must be defined in the list of OSS NFS user IDs. If FALSE, a user must have an alias to gain access through pass-through servers. The default value is FALSE.

`PRI` *`cpu-priority`*

specifies the execution priority for the server process. You can specify a priority of -1 or a priority from 1 through 199. (Processes with higher numbers have higher priority.) If you specify a priority of -1 or omit this attribute, the priority of the NFS manager process at the time the SERVER object is started is used as the priority of the server process.

`PROGRAM` *`file-name`*

specifies the NonStop Kernel file name of the server program. Use the filename that was specified when the NFSSVRHP file was installed. See the *Open System Services NFS Management and Operations Guide* for more information about installation.

`PROCESS` *`process-name`*

specifies the name of the server process.

`READ-ONLY { TRUE | FALSE }`

specifies whether or not access to the file system serviced by the server is on a read-only or read/write basis. If TRUE, the file system must be initialized before you create this server by a server that is not configured as read-only. The default value is FALSE—the file system allows read/write access.

`ROOT-USER-OK { TRUE | FALSE }`

specifies whether remote root users are allowed to access the server’s files with super-ID privileges. Root users have access to all NFS files, regardless of their permission and ownership values. This super-ID permission is normally not allowed during access checking because anyone who can become the root user on their workstation can gain access to any remote file. If TRUE, remote root users are allowed access to the server’s files with super-ID privileges. If FALSE, requests from remote root users are treated as requests from the NFS nobody user (user ID 65534). Note that the FALSE value is contingent on the definition of user ID 65534 as the nobody user ID. To prevent super-ID permission during access checking, a nobody user ID must be defined.

The default value is FALSE.

`STYPE OSS`

specifies that the type of server is an OSS NFS file server. NFS file servers store and provide access to NFS files, which are typically in formats compatible with PCs or workstations: that is, files compatible with DOS or UNIX.

`SWAP [` *`file-name`* `]`

specifies a file used to hold the server process's virtual data. This attribute allows you to specify a permanent file for swapping the data stack and to specify a different volume for the swap file. See the RUN command in the *TACL Reference Manual* for more information about specifying a swap file.

`WRITE-THRU { TRUE | FALSE }`

specifies whether data written by NFS clients to NonStop Himalaya systems is written to cache or directly to disk. Configurations involving high-performance applications should set this parameter to TRUE while those involving high-reliability applications should set this parameter to false.

The default value is FALSE.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL STOPPED option to specify that you want to alter only SERVER objects in the STOPPED summary state.

Considerations

Consider the following points when using the ALTER SERVER command:

- ALTER SERVER is a sensitive command requiring super-group access, or super-ID access if NULL-ALIAS-OK and ROOT-USER-OK are TRUE.
- When you alter a SERVER object, it must be in the STOPPED summary state.
- To remove the existing SWAP attribute value, specify the SWAP attribute keyword without a filename value.
- Use the INFO SERVER command to determine the current attribute values of a SERVER object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.
- To move a backup server process from one processor to another, two ALTER SERVER commands are required. The current backup process must first be stopped by issuing an ALTER SERVER command that sets the BACKUP attribute to -1. The backup process can then be restarted in the target process by issuing an ALTER SERVER command that sets the BACKUP attribute to the value of the target processor.

Example

The following command alters the local mount for a server named LIB to be the `/usr/lib` directory and specifies a backup processor for the server process:

```
->ALTER SERVER $ZNFS.LIB, MNTPOINT /usr/lib, BACKUP 3
```

ALTER SUBSYS Command

The ALTER SUBSYS command changes attribute values associated with the specified SUBSYS object.

The SUBSYS object is used to define the OSS NFS subsystem as a whole. There is only one SUBSYS object per OSS NFS subsystem.

The ALTER SUBSYS command has the following syntax:

```
ALTER SUBSYS process-name [ [ , attribute-spec ]... ]

      [ , SEL [ NOT ] state ]
```

For a SUBSYS object, *attribute-spec* can be:

```
REUSE-SERVERID { ON | OFF }
```

`SUBSYS` *`process-name`*

specifies the name of the SUBSYS object whose attributes are to be altered. The SUBSYS object uses the name of the manager process with which it is associated.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to alter only SUBSYS objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

`REUSE-SERVERID { ON | OFF }`

specifies whether the server IDs assigned when servers are started initially are to be reused after the servers are stopped and restarted. Changing a server ID causes the corresponding file system to be unavailable to any clients that still have remote mounts against the server. Any file handles held by these clients become stale (file handle points to a file that no longer exists or is no longer accessible). Allowing the server ID to be reused enables the client to continue using a server without remounting it after it is stopped and restarted. The default value is ON.

Considerations

Consider the following points when using the ALTER SUBSYS command:

- ALTER SUBSYS is a sensitive command requiring super-group access.

- Use the STATUS SUBSYS command to determine the current subsystem attribute values.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. For the SUBSYS object, use the STATUS SUBSYS command to determine which of the requested changes were made before the error occurred.
- Normally, to prevent disruption to NFS clients, the REUSE-SERVERID attribute should be left ON. If an important server attribute (such as its mount point) is changed, and the files maintained by that server really are different, then REUSE-SERVERID can be turned OFF to make sure that any clients with remote mounts against the server are not unintentionally given access to the wrong files. The following example shows how the ALTER SUBSYS command can be used among a sequence of commands to first turn off and then turn on the REUSE-SERVERID attribute in just such a case:

```
-> ASSUME $ZNFS
-> STOP SERVER ROOT
-> ALTER SERVER ROOT, MNTPOINT
-> ALTER SUBSYS, REUSE-SERVERID OFF
-> START SERVER ROOT
-> ALTER SUBSYS, REUSE-SERVERID ON
```

Example

The following command disallows reuse of the server ID for the SUBSYS object associated with the manager process $ZNFS:

```
->ALTER SUBSYS $ZNFS, REUSE-SERVERID OFF
```

ALTER USER Command

The ALTER USER command changes attribute values associated with the specified USER object.

The ALTER USER command has the following syntax:

```
ALTER USER user-name [ [ , attribute-spec ]... ]
```

For a USER object, *attribute-spec* can be bany of:

```
[ ALIAS OSS { user-name   }
            { user-number } ]
[ COMMENT “string” ]
```

USER *user-name*

specifies the name of the USER object whose attributes are to be altered.

```
ALIAS OSS { user-name   }
          { user-number }
```

specifies a NonStop Kernel user ID, referred to as the mapped user ID, to be used by this NFS user when requesting file access. The NonStop Kernel user ID must already be defined in the USERID file on the system on which the OSS NFS manager process is running. *`user-name`* must be a NonStop Kernel user name. If you omit the ALIAS OSS attribute, the user has no mapped user ID.

`COMMENT` *“string”*

is an optional attribute that allows you to include information about the user. The information is in readable ASCII format and can be from 1 through 128 characters long. For example, you might use this attribute to record the user’s full name and office extension.

Considerations

Consider the following points when using the ALTER USER command:

- ALTER USER is a sensitive command requiring super-ID access.
- To remove an existing mapped user ID, specify the ALIAS OSS attribute without a user specification. To remove an existing user comment, specify the COMMENT attribute with an empty string.
- To alter a USER object, you must be the super ID.
- Use the INFO USER command to determine the current attribute values of a USER object.
- If you specify multiple parameters in any ALTER command and an error occurs, some of the processing might be complete even though the processing was interrupted. Use the appropriate INFO command to determine which of the requested changes were made before the error occurred.
- GROUPID and USERID cannot be modified with the ALTER USER command.

Example

The following command alters the OSS mapped user ID of a user named phil:

```
->ALTER USER phil, ALIAS OSS sales_mgr
```

DELETE EXPORT Command

The DELETE EXPORT command removes specified EXPORT objects from the OSS NFS subsystem. The deletion of an EXPORT object prevents the granting of new remote mounts against the exported file system, but does not affect the status of remote mounts that have already been granted.

The DELETE EXPORT command has the following syntax:

```
DELETE EXPORT export-name
```

`EXPORT` *`export-name`*

specifies the name of the EXPORT object to be deleted.

Considerations

Consider the following points when using the DELETE EXPORT command:

- DELETE EXPORT is a sensitive command requiring super-ID access.

Example

The following command deletes an EXPORT object named labs:

```
->DELETE EXPORT $ZNFS.labs
```

DELETE GROUP Command

The DELETE GROUP command removes specified GROUP objects from the OSS NFS subsystem.

The DELETE GROUP command has the following syntax:

```
DELETE GROUP group-name
```

`GROUP` *`group-name`*

specifies the name of the GROUP object to be deleted.

Considerations

Consider the following points when using the DELETE GROUP command:

- DELETE GROUP is a sensitive command requiring super-group access.
- The DELETE GROUP command does not delete the USER objects that are members of the group. You delete USER objects by using the DELETE USER command.

Example

The following command deletes a group named writers:

```
->DELETE GROUP $ZNFS.writers
```

DELETE LAN Command

The DELETE LAN command removes specified LAN objects from the OSS NFS subsystem.

The DELETE LAN command has the following syntax:

```
DELETE LAN lan-name [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object to be deleted.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to delete only LAN objects in the STOPPED summary state.

Considerations

Consider the following points when using the DELETE LAN command:

- DELETE LAN is a sensitive command requiring super-group access.
- When you delete a LAN object, it must be in the STOPPED summary state.

Example

The following command deletes a LAN object named LAN0:

```
->DELETE LAN $ZNFS.LAN0
```

DELETE NETGROUP Command

The DELETE NETGROUP command removes specified NETGROUP objects from the OSS NFS subsystem.

The DELETE NETGROUP command has the following syntax:

```
DELETE NETGROUP netgroup-name
```

`NETGROUP` *`netgroup-name`*

specifies the name of the NETGROUP object to be deleted.

Considerations

Consider the following points when using the DELETE NETGROUP command:

- DELETE NETGROUP is a sensitive command requiring super-ID access.

- The DELETE NETGROUP command does not delete the NETGROUP or USER objects that are members of the netgroup. Each NETGROUP object must be specifically deleted. You delete USER objects by using the DELETE USER command.

Example

The following command deletes a netgroup named labtech:

```
->DELETE NETGROUP $ZNFS.labtech
```

DELETE SERVER Command

The DELETE SERVER command removes specified SERVER objects from the OSS NFS subsystem.

The DELETE SERVER command has the following syntax:

```
DELETE SERVER server-name [ , SEL [ NOT ] state ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object to be deleted.

`SEL [ NOT ]` *`state`*

specifies the SEL option. You can use the SEL option to specify that you want to delete only SERVER objects in the STOPPED summary state.

Considerations

Consider the following points when using the DELETE SERVER command:

- DELETE SERVER is a sensitive command requiring super-group access or super-ID access if NULL-ALIAS-OK and ROOT-USER-OK are TRUE.
- When you delete a SERVER object, it must be in the STOPPED summary state.

Example

The following command deletes a server named ROOT (but only if it is in the STOPPED summary state):

```
->DELETE SERVER $ZNFS.ROOT, SEL STOPPED
```

DELETE USER Command

The DELETE USER command removes specified USER objects from the OSS NFS subsystem.

The DELETE USER command has the following syntax:

```
DELETE USER user-name
```

`USER` *`user-name`*

specifies the name of the USER object to be deleted.

Considerations

Consider the following points when using the DELETE USER command:

- DELETE USER is a sensitive command requiring super-ID access.
- Deleting a USER object does not remove the object from the membership list of any groups.
- To delete a USER object, you must be the super-user.

Example

The following command deletes a user named shirley:

```
->DELETE USER $ZNFS.shirley
```

INFO EXPORT Command

The INFO EXPORT command displays the current attribute values for the specified EXPORT objects.

You can display attribute information about EXPORT objects with or without detail.

The information contained in the displays produced by INFO EXPORT is described in detail under Examples on page 2-42.

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO EXPORT command has the following syntax:

```
INFO EXPORT export-name [ , DETAIL ]
```

`EXPORT` *`export-name`*

specifies the name of the EXPORT object for which attribute information is requested.

`DETAIL`

specifies that detailed EXPORT attribute information is requested.

Considerations

Consider the following points when using the INFO EXPORT command.

- INFO EXPORT is a nonsensitive command.
- The wildcard character for INFO EXPORT is **, not *.

Examples

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about all currently defined EXPORT objects:

```
->INFO EXPORT $ZNFS.**
```

The resulting display has the following format:

```
NFS Info EXPORT \SYS1.$ZNFS.**

Objname           *Access
/                  everyone
/usr               pcsys, tsys1
/docs              medaccts, techinfo
```

The display contains the following information:

Objname	is the path name of the directory being exported.
Access	is a list of hosts or netgroups that may access files in or below the exported directory. If no names appear, the directory and all files below it in the hierarchy can be accessed by all hosts.

The following command displays detailed information about the EXPORT object `/usr:`

```
->INFO EXPORT $ZNFS./usr, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Info EXPORT \SYS1.$ZNFS./usr

*Access..........  pcsys
*Access..........  tsys1
```

The detailed display contains the following information:

Access	is a list of hosts or netgroups that may access files in or below the exported directory. If no names appear, the directory and all files below it in the hierarchy can be accessed by all hosts.

INFO GROUP Command

The INFO GROUP command displays the current attribute values for the specified GROUP objects.

You can display attribute information about GROUP objects with or without detail.

The information contained in the displays produced by INFO GROUP is described in detail under Examples on page 2-44.

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO GROUP command has the following syntax:

```
INFO GROUP group-name [ , GROUPID group-id ] [ , DETAIL ]
```

`GROUP` *`group-name`*

specifies the name of the GROUP object for which attribute information is requested.

`GROUPID` *`group-id`*

specifies the group number of the GROUP object for which attribute information is requested. If specified, GROUP objects whose names match the group-name specification must also have a group number matching the GROUPID value for information about them to be returned. This attribute is usually used with an object-name template when the group number but not the group name is known.

`DETAIL`

specifies that detailed GROUP attribute information is requested.

Considerations

Consider the following points when using the INFO GROUP command.

- INFO GROUP is a nonsensitive command.

Examples

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about all currently defined GROUP objects:

```
->INFO GROUP $ZNFS.*
```

The resulting display has the following format:

```
NFS Info GROUP \SYS1.$ZNFS.*

Objname          *GID  *Members
sales             50   alan, janice, susie, phil
software          25   garyt, garys, marilyn
```

The display contains the following information:

Objname	is the name of the group.
GID	is the NFS group ID corresponding to the group name.
Members	is a list of NFS users who are members of the group.

The following command displays detailed information about the GROUP object named sales:

```
->INFO GROUP $ZNFS.sales, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Info GROUP \SYS1.$ZNFS.sales

*Group ID......... 25
*Member........... alan
*Member........... janice
*Member........... susie
*Member........... phil
```

The detailed display contains the following information:

Group ID	is the NFS group ID corresponding to the group name.
Member	is a member of the group.

INFO LAN Command

The INFO LAN command displays the current attribute values for the specified LAN objects.

You can display attribute information about LAN objects with or without detail.

The information contained in the displays produced by INFO LAN is described in detail under Examples on page 2-45.

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO LAN command has the following syntax:

```
INFO LAN lan-name [ , DETAIL ] [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object for which attribute information is requested.

`DETAIL`

specifies that detailed LAN attribute information is requested.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you are requesting attribute information about only LAN objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the INFO LAN command.

- INFO LAN is a nonsensitive command.
- Use the STATUS LAN command to obtain information about the current state of a LAN object.
- Use the STATISTICS LAN command to obtain statistical information about a LAN object.

Examples

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about a LAN object named NFSLAN:

```
->INFO LAN $ZNFS.NFSLAN
```

The resulting display has the following format:

```
NFS Info LAN \SYS1.$ZNFS.NFSLAN
Objname  *Process  *Primary *Backup *Priority *TCP/IP
                                               Process
NFSLAN   $NFSL        1       -1       148     $ZTC0
```

The display contains the following information:

Objname	is the LAN object name.
Process	is the name of the LAN interface process.
Primary	is the number of the primary processor.
Backup	is the number of the backup processor. A value of -1 indicates no backup processor is defined and the LAN interface process will not be restarted in case of failure.
Priority	is the execution priority for the LAN interface process. If the priority value is followed by the (DEFAULT) indicator, the LAN object is configured to use the same priority as the manager process, and the indicated value is the current manager process priority.
TCP/IP Process	is the name of the TCP/IP process used by the LAN interface process.

The following command displays detailed information about the same LAN object:

```
->INFO LAN $ZNFS.NFSLAN, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Info LAN \SYS1.$ZNFS.NFSLAN

*Process.......... $NFSL
*Program.......... $SYSTEM.ZOSSNFS.NFSLAN
*Swap............. $SWAP
*TCP/IP Process... $ZTC0
*TCP/IP Host File. $SYSTEM.ZTCPIP.HOSTS
*TCP/IP Resolver.. $SYSTEM.ZTCPIP.RESCONF
*Primary.......... 1       *Backup ......... -1
*Priority......... 148 (DEFAULT)
*Histogram........ 128, 256, 512, 1024, 2048, 4096
*Addr Check....... ON
*Domain Name...... sitexyz.com
```

The detailed display contains the following information:

Process	is the name of the LAN interface process.
Program	is the file name of the LAN interface program.
Swap	is the swap volume or swap file to be used when the LAN interface process is started.
TCP/IP Process	is the name of the TCP/IP process used by the LAN interface process.
TCP/IP Host File	is the name of the NonStop TCP/IP HOSTS file. If not specified, NonStop TCP/IP uses a domain name resolver.
TCP/IP Resolver	is the name of the NonStop TCP/IP resolver file. NonStop TCP/IP ignores this attribute if a HOSTS file is specified. If neither a

	HOSTS file, nor a resolver file is specified, NonStop TCP/IP uses the file $SYSTEM.ZTCPIP.RESCONF.
Primary	is the number of the primary processor.
Backup	is the number of the backup processor. A value of -1 indicates no backup processor is defined and the LAN interface process will not be restarted in case of failure.
Priority	is the execution priority for the LAN interface process. If the priority value is followed by the (DEFAULT) indicator, the LAN object is configured to use the same priority as the manager process, and the indicated value is the current manager process priority.
Histogram	is a list of the maximum lengths in bytes of messages counted in each buffer used to produce the histogram statistics displayed by the STATS LAN command.
Addr Check	indicates whether the host name is checked (ON) or not checked (OFF) when a mount or unmount request is received on the LAN interface process.
Domain Name	is the domains that correspond to the domain names specified when defining NETGROUP objects. See ADD GROUP Command on page 2-11 for information on the use of domains.

INFO NETGROUP Command

The INFO NETGROUP command displays the current attribute values for the specified NETGROUP object(s).

The information contained in the displays produced by INFO NETGROUP is described in detail under “Examples.”

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO NETGROUP command has the following syntax:

```
INFO NETGROUP netgroup-name
```

`NETGROUP` *`netgroup-name`*

specifies the name of the NETGROUP object for which attribute information is requested.

Considerations

Consider the following points when using the INFO NETGROUP command.

- INFO NETGROUP is a nonsensitive command.

- The DETAIL option is not allowed with the INFO NETGROUP command.

Example

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about a netgroup named medaccts:

```
->INFO NETGROUP $ZNFS.medaccts
```

The resulting display has the following format:

```
NFS Info NETGROUP \SYS1.$ZNFS.medaccts

*Member.......... (sunsys, marilyn, techco)
*Member.......... (tsys1, garyt, techco)
*Member.......... bldg3_labs
```

The display contains the following information:

Member	is a member of the netgroup, which can be another netgroup or a list enclosed in parentheses that indicates a host name, user name, and domain name.

INFO PROCESS Command

The INFO PROCESS command displays the current attribute values for the specified PROCESS object. When combined with the SUB and SEL options, this command displays the current attribute information for all objects that meet the specified *process-name*, SUB, and SEL specifications.

You can display attribute information about the specified objects with or without detail.

The information contained in the displays produced by INFO PROCESS is described in detail under “Examples.”

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO PROCESS command has the following syntax:

```
INFO PROCESS process-name [ ,  DETAIL  ]

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

     [ , SEL [ NOT ] state ]
```

`PROCESS` *`process-name`*

specifies the name of the PROCESS object for which attribute information is requested.

`DETAIL`

specifies that detailed attribute information is requested.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can specify SUB ALL to request information about the PROCESS object and all objects subordinate to the PROCESS object, SUB ONLY to request information about all objects subordinate to the PROCESS object, or specify one of the object types to request information about that object.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you are requesting attribute information about only objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the INFO PROCESS command.

- INFO PROCESS is a nonsensitive command.
- Use the STATUS PROCESS command to obtain information about the current state of the PROCESS object.
- Use the STATISTICS PROCESS command to obtain statistical information about the PROCESS object.
- The information contained in the displays produced for subordinate objects is described under the individual command descriptions for each object.

Examples

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about the manager process $ZNFS:

```
->INFO PROCESS $ZNFS
```

The resulting display has the following format:

```
NFS Info PROCESS \SYS1.$ZNFS

Objname  Primary  *Backup  *BackupDebug  *DebugOnError  *Collector
$ZNFS      10       -1         OFF            OFF         \SYS1.$0
```

The display contains the following information:

Objname	is the PROCESS object name.
Primary	is the number of the primary CPU.
Backup	is the number of the backup processor. A value of -1 indicates no backup processor is defined and the NFS manager process is not being run as a process pair.
Backup Debug	indicates whether the backup process should (ON) or should not (OFF) be initiated in debug mode.
Debug On Error	indicates whether the program enters (ON) or does not enter (OFF) debug mode when a fatal error occurs.
Collector	is the name of the EMS collector file.

The following command displays detailed information about the manager process $ZNFS:

```
->INFO PROCESS $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Info PROCESS \SYS1.$ZNFS

 Program.......    \SYS1.$SYSTEM.ZOSSNFS.NFSMGR
 Program Mod Time. 30 Mar 1995, 16:33:24.110
*Msgfile.......... NONE
*Collector........ \SYS1.$0
*Logfile.......... NONE
 Config Subvol.... \SYS1.$SYSTEM.ZOSSNFS
 Data Swap Vol.... $SYSTEM              Ext Mem Swap Vol.  $SYSTEM
 Primary.......... 0                   *Backup...........  1
*Backup Debug..... OFF                 *Debug On Error...  OFF
```

The detailed display contains the following information:

Program	is the file name of the manager program.
Program Mod Time	is the time the program file was last modified (modification timestamp).
Msgfile	is the name of the message template file used by the manager process to generate text messages, if any.
Msgfile Mod Time	is the time the message template file was last modified, if this file is defined.
Collector	is the name of the EMS collector file.
Logfile	is the name of the manager process log file. The text corresponding to any events generated by the manager process is written to this file.
Data Swap Vol	is the swap volume used for the process data stack (as specified in the SWAP parameter of the RUN command).

Ext Mem Swap Vol	is the swap volume used for the process extended memory segment (as specified in the SWAPVOL parameter of the PARAM or RUN command).
Primary	is the number of the primary processor.
Backup	is the number of the backup processor. A value of -1 indicates no backup processor is defined and the NFS manager process is not being run as a process pair.
Backup Debug	indicates whether the backup process should (ON) or should not (OFF) be initiated in debug mode.
Debug On Error	indicates whether the program enters (ON) or does not enter (OFF) debug mode when a fatal error occurs.

INFO SERVER Command

The INFO SERVER command displays the current attribute values for the specified SERVER objects.

You can display attribute information about SERVER objects with or without detail.

The information contained in the displays produced by INFO SERVER is described in detail under “Examples on page 2-52.

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO SERVER command has the following syntax:

```
INFO SERVER server-name [ ,  DETAIL  ]

     [ , SEL [ NOT ] state ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object for which attribute information is requested.

`DETAIL`

specifies that detailed SERVER attribute information is requested.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you are requesting attribute information about only SERVER objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the INFO SERVER command.

- INFO SERVER is a nonsensitive command.
- Use the STATUS SERVER command to obtain information about the current state of a SERVER object.
- Use the STATISTICS SERVER command to obtain statistical information about a SERVER object.

Examples

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about all currently defined SERVER objects:

```
->INFO SERVER $ZNFS.*
```

The resulting display has the following format:

```
NFS Info SERVER \SYS1.$ZNFS.*

Objname *Process *Primary *Backup   *Priority *Mount Point
SRV1    $SRV1       0        -1         148     /
PSRV2   $PSRV2      1        -1         148     /docs
NSRV3   $NSRV      0          1         148     /usr
```

The display contains the following information:

Objname	is the SERVER object name.
Process	is the name of the server process.
Primary	is the number of the primary processor.
Backup	is the number of the backup processor. A value of -1 indicates no backup processor is defined and the server process is not being run in persistent mode.
Priority	is the execution priority for the server process. If the priority value is followed by the (DEFAULT) indicator, the SERVER object is configured to use the same priority as the manager process, and the indicated value is the current manager process priority.
Mount Point	is the pathname of the directory file on which the server’s file system is mounted locally.

The following command displays detailed information about the server named ROOT:

```
->INFO SERVER $ZNFS.ROOT, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Info SERVER \SYS1.$ZNFS.ROOT

*Mount Point.... /
*Program........ $SYSTEM.ZOSSNFS.NFSSVRHP
*Process........ $SV001      *Server Type...... OSS
 Primary........ 1           *Backup........... 0
*Read Only...... FALSE       *Priority......... 148
*Root User OK... TRUE        *Max File Size.... 2147483647
*Write Thru..... TRUE        *Null Alias OK.... FALSE
```

The detailed display contains the following information:

Mount Point	is the pathname of the directory file on which the server's file system is mounted locally.
Program	is the NonStop Kernel filename of the server program.
Process	is the name of the server process.
Server Type	is the type of server.
Primary	is the number of the primary processor.
Backup	is the number of the backup processor. A value of -1 indicates no backup processor is defined and the server process is not being run in persistent mode.
Read Only	indicates whether all files and directories in the file system allow only read access (TRUE) or allow other types of access (FALSE).
Priority	is the execution priority for the server process. If the priority value is followed by the (DEFAULT) indicator, the SERVER object is configured to use the same priority as the manager process, and the indicated value is the current manager process priority.
Root User OK	indicates whether remote users are allowed to access (TRUE) or cannot access (FALSE) the server's files with super-ID privileges.
Max File Size	is the maximum number of bytes for any given file.
Write Thru	indicates whether data written by NFS clients to NonStop Himalaya systems is written to cache or directly to disk.
Null Alias OK	indicates whether remote NFS users who do not have a mapped user ID are allowed access to files managed by the NonStop system (TRUE) or are not allowed this access (FALSE).

INFO USER Command

The INFO USER command displays the current attribute values for the specified USER objects.

You can display attribute information about USER objects with or without detail.

The information contained in the displays produced by INFO USER is described in detail under Examples on page 2-55.

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The INFO USER command has the following syntax:

```
INFO USER user-name [ , USERID user-id ]

     [ , GROUPID group-id ] [ ,  DETAIL  ]
```

`USER` *`user-name`*

specifies the name of the USER object for which configuration information is requested.

`USERID` *`user-id`*

specifies the user number of the USER object for which attribute information is requested. If specified, USER objects whose names match the user-name specification must also have a user number matching the USERID value for information about them to be returned. This attribute can be used with an object name template when the user number but not the user name is known.

`GROUPID` *`group-id`*

specifies the group number of the USER object for which attribute information is requested. If specified, USER objects whose names match the user-name specification must also have a group number matching the GROUPID value for information about them to be returned. This attribute can be used with an object name template to look up all the users with a specific group number.

`DETAIL`

specifies that detailed USER attribute information is requested.

Considerations

Consider the following point when using the INFO USER command:

- INFO USER is a nonsensitive command.

Examples

Note. An asterisk (*) in any INFO command display indicates that you can change the value of the attribute using the ALTER command.

The following command displays information (without detail) about all currently defined USER objects:

```
->INFO USER $ZNFS.*
```

The resulting display has the following format:

```
NFS Info USER \SYS1.$ZNFS.*

Objname   *UID   *GID  *Alias Name        Type    *Comment
carolyn   1263    247   SOFTWARE.CAROLYN  OSS
user         8    247   SOFTWARE.USER     OSS
```

The display contains the following information:

Objname	is the name of the USER object.
UID	is the NFS user ID of the user.
GID	is the NFS group ID of the user, if any.
Alias Name	is the mapped NonStop Kernel user name defined for the NFS user.
Type	specifies the type of mapped user ID. The only value for this field can be OSS.
Comment	is descriptive information about the user.

The following command displays detailed information about the user named susie:

```
->INFO USER $ZNFS.susie, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Info USER \SYS1.$ZNFS.guest

*User ID........ 8                *Group ID...... 247
*Comment........ Guest user id
*OSS Name....... SOFTWARE.USER    *OSS ID........ 510
```

The detailed display contains the following information:

User ID	is the NFS user ID of the user.
Group ID	is the NFS group ID of the user, if any.
Comment	is descriptive information about the user.
OSS Name	is the NonStop Kernel user name that maps to the NFS user.
OSS ID	is the NonStop Kernel user ID that maps to the NFS user.

LISTOPENS PROCESS Command

The LISTOPENS PROCESS command returns a list of file system opens associated with the NFS manager process.

The information contained in the displays produced by LISTOPENS PROCESS is described in detail under Example on page 2-56.

The LISTOPENS PROCESS command has the following syntax:

```
LISTOPENS PROCESS process-name
```

PROCESS *process-name*

specifies the name of the PROCESS object for which open information is requested.

Considerations

Consider the following point when using the LISTOPENS PROCESS command:

- LISTOPENS PROCESS is a nonsensitive command.

Example

The following command lists the file system opens associated with the NFS manager process:

```
->LISTOPENS PROCESS $ZNFS
```

The resulting display has the following format:

```
NFS Listopens PROCESS \SYS1.$ZNFS

Opener          PPID       BPID       Qualifier
\SYS1.$LAN0     1,33                  #ZRPC
\SYS1.$ZNET     1,31                  #ZSPI
```

The display contains the following information:

Opener	is the process name of the opener.
PPID	is the process ID of the primary opener.
BPID	is the process ID of the backup opener. In this example, there is no BPID defined.
Qualifier	is the process name qualifier used by the opener to open the manager process.

LISTOPENS SUBSYS Command

The LISTOPENS SUBSYS command returns a list of remote mounts associated with the OSS NFS subsystem.

You can list the remote mounts with or without detail.

The information contained in the displays produced by LISTOPENS SUBSYS is described in detail under Examples on page 2-57.

The LISTOPENS SUBSYS command has the following syntax:

```
LISTOPENS SUBSYS process-name [ , DETAIL ]
```

`SUBSYS` *`process-name`*

specifies the name of the SUBSYS object for which remote mount information is requested. The SUBSYS object uses the name of the manager process with which it is associated.

`DETAIL`

specifies that the more detailed remote mount listing is requested.

Considerations

Consider the following points when using the LISTOPENS SUBSYS command:

- LISTOPENS SUBSYS is a nonsensitive command.
- You can use the LISTOPENS SUBSYS command to verify that all remote mounts have been unmounted before you use the STOP SUBSYS command. However, be aware that the remote mount information returned might not be accurate or current due to broken connections, takeovers, and system failures that cause information to be lost or become unsynchronized (out of sync). For example, if an NFS client’s system fails and is rebooted, the rebooted system might be unaware that a remote mount had existed. The NFS client will no longer have a remote mount, but the LISTOPENS display will show the client as a remote mount owner because OSS NFS has not received an UNMOUNT request.

Examples

The following command lists the remote mounts associated with the OSS NFS subsystem that has a manager process named $ZNFS:

```
->LISTOPENS SUBSYS $ZNFS
```

The resulting display has the following format:

```
NFS Listopens SUBSYS \SYS1.$ZNFS

Host                Mount Point
pcsys               /
pcsys               /usr
sunsys              /docs
```

The display contains the following information:

Host	is the machine name of a host system that currently has one or more remote mounts against a part of the OSS NFS file hierarchy.
Mount Point	is the pathname of a directory that has been remotely mounted by the host.

The following command lists in detail the remote mounts associated with the OSS NFS subsystem that has a manager process named $ZNFS:

```
->LISTOPENS SUBSYS $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Listopens SUBSYS

Host Name           pcsys
Mount Point         /
                    /usr

Host Name           sunsys
Mount Point         /docs
```

The detailed display contains the following information:

Host Name	is the machine name of a host system that currently has one or more remote mounts against a part of the OSS NFS file hierarchy.
Mount Point	is the pathname of a directory that has been remotely mounted by the host.

NAMES EXPORT Command

The NAMES EXPORT command displays the names of all EXPORT objects that meet the *`export-name`* specification.

This command is most useful if you specify an object-name template. See Object-Name Templates on page 1-6.

The NAMES EXPORT command has the following syntax:

```
NAMES EXPORT export-name
```

`EXPORT` *`export-name`*

specifies the name of the EXPORT objects to be returned.

Considerations

Consider the following points when using the NAMES EXPORT command:

- NAMES EXPORT is a nonsensitive command.
- Because some EXPORT object names can be up to 1024 characters long, names longer than the record length of the SCF output device are wrapped into multiple lines.
- The wildcard character for NAMES EXPORT is **, not *.

Example

The following command displays the names of all EXPORT objects associated with the manager process $ZNFS:

```
->NAMES EXPORT $ZNFS./**
```

The resulting display has the following format:

```
NFS Names EXPORT \SYS1.$ZNFS./*

EXPORT
/acct  /etc   /man   /usr
```

NAMES GROUP Command

The NAMES GROUP command displays the names of all GROUP objects that meet the *`group-name`* specification.

This command is most useful if you use an object-name template. See Object-Name Templates on page 1-6.

The NAMES GROUP command has the following syntax:

```
NAMES GROUP group-name
```

`GROUP` *`group-name`*

specifies the name of the GROUP objects to be returned.

Considerations

Consider the following point when using the NAMES GROUP command:

- NAMES GROUP is a nonsensitive command.

Example

The following command displays the names of all GROUP objects associated with the manager process $ZNFS:

```
->NAMES GROUP $ZNFS.*
```

The resulting display has the following format:

```
NFS Names GROUP \SYS1.$ZNFS.*

GROUP
bin    nobody    software   turtle
```

NAMES LAN Command

The NAMES LAN command displays the names of all LAN objects that meet the *lan-name* and SEL specifications.

This command is most useful if you use an object-name template. See Object-Name Templates on page 1-6.

The NAMES LAN command has the following syntax:

```
NAMES LAN lan-name [ , SEL [ NOT ] state ]
```

`LAN` *lan-name*

specifies the name of the LAN objects to be returned.

`SEL [ NOT ]` *state*

specifies the summary state select option. You can use the SEL option to specify that you are requesting attribute information about only LAN objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following point when using the NAMES LAN command:

- NAMES LAN is a nonsensitive command.

Example

The following command displays the names of all LAN objects in the STARTED summary state:

```
->NAMES LAN $ZNFS.*, SEL STARTED
```

The resulting display has the following format:

```
NFS Names LAN \SYS1.$ZNFS.*

LAN
NFSLAN
```

NAMES NETGROUP Command

The NAMES NETGROUP command displays the names of all NETGROUP objects that meet the `netgroup-name` specification.

This command is most useful if you use wild-card characters in the object name you specify. For example, you can use an asterisk (*) in place of the name to specify all objects of the type you request. See “Object Names,” in Section 1, for a description of how to use wild-card characters.

The NAMES NETGROUP command has the following syntax:

```
NAMES NETGROUP netgroup-name
```

`NETGROUP` *`netgroup-name`*

specifies the name of the NETGROUP objects to be returned.

Considerations

Consider the following point when using the NAMES NETGROUP command:

- NAMES NETGROUP is a nonsensitive command.

Example

The following command displays the names of all netgroups associated with the $ZNFS manager process:

```
->NAMES NETGROUP $ZNFS.*
```

The resulting display has the following format:

```
NFS Names NETGROUP \SYS1.$ZNFS.*

NETGROUP
bldg3_labs       labs
```

NAMES PROCESS Command

The NAMES PROCESS command displays the names of all objects that meet the *`process-name`*, SUB, and SEL specifications. This command is most useful when combined with the SUB and SEL options.

The NAMES PROCESS command has the following syntax:

```
NAMES PROCESS process-name

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

      [ , SEL [ NOT ] state ]
```

`PROCESS` *`process-name`*

specifies the name of the PROCESS object to which the command applies.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can use the SUB option to specify that you want to return the names of only specified subordinate objects.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to return the names of only objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the NAMES PROCESS command:

- NAMES PROCESS is a nonsensitive command.
- Because OSS NFS object names can be up to 1024 characters long, names longer than the record length of the SCF output device are wrapped into multiple lines.

Example

The following command displays the names of the $ZNFS PROCESS object and all subordinate OSS NFS objects in the STARTED summary state:

```
->NAMES PROCESS $ZNFS, SUB ALL, SEL STARTED
```

The resulting display has the following format:

```
NFS Names PROCESS \SYS1.$ZNFS

PROCESS
$ZNFS

LAN
NFSLAN

SERVER
ROOT   SERVER2   SERVER3
```

NAMES SERVER Command

The NAMES SERVER command displays the names of all SERVER objects that meet the *server-name* and SEL specifications.

This command is most useful if you use an object-name template. See Object-Name Templates on page 1-6.

The NAMES SERVER command has the following syntax:

```
NAMES SERVER server-name [ , SEL [ NOT ] state ]
```

SERVER *server-name*

specifies the name of the SERVER objects to be returned.

SEL [NOT] *state*

specifies the summary state select option. You can use the SEL option to specify that you want to return the names of only SERVER objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the NAMES SERVER command:

- NAMES SERVER is a nonsensitive command.

Example

The following command displays the names of all servers in the STARTED summary state:

```
->NAMES SERVER $ZNFS.*, SEL STARTED
```

The resulting display has the following format:

```
NFS Names SERVER \SYS1.$ZNFS.*

SERVER
ROOT      USR       GRD
```

NAMES SUBSYS Command

The NAMES SUBSYS command displays the names of the SUBSYS objects that meet the *process-name*, SUB, and SEL specifications. This command is most useful when combined with the SUB and SEL options.

The NAMES SUBSYS command has the following syntax:

```
NAMES SUBSYS process-name

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

      [ , SEL [ NOT ] state ]
```

`SUBSYS` *`process-name`*

specifies the name of the SUBSYS object to which the command applies. The SUBSYS object uses the name of the manager process with which it is associated.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can use the SUB option to specify that you want to return the names of only specified subordinate objects.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to return the names of only objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the NAMES SUBSYS command:

- NAMES SUBSYS is a nonsensitive command.
- Because OSS NFS object names can be up to 1024 characters long, names longer than the record length of the SCF output device are wrapped into multiple lines.

Example

This command displays the name of the SUBSYS object associated with the manager process $ZNFS, and all subordinate OSS NFS objects in the STARTED summary state:

```
->NAMES SUBSYS $ZNFS, SUB ALL, SEL STARTED
```

The resulting display has the following format:

```
NFS Names SUBSYS \SYS1.$ZNFS

SUBSYS
$ZNFS

PROCESS
$ZNFS

LAN
NFSLAN

SERVER
ROOT   SERVER2   SERVER3
```

NAMES USER Command

The NAMES USER command displays the names of all USER objects that meet the *user-name* specification.

This command is most useful if you use an object-name template. See "Object-Name Templates on page 1-6.

The NAMES USER command has the following syntax:

```
NAMES USER user-name
```

`USER` *`user-name`*

specifies the name of the USER objects to be returned.

Considerations

Consider the following point when using the NAMES USER command:

- NAMES USER is a nonsensitive command.

Example

The following command displays the names of all users associated with the $ZNFS process:

```
->NAMES USER $ZNFS.*
```

The resulting display has the following format:

```
NFS Names USER \SYS1.$ZNFS.*

USER
donatello   leonardo   michelangelo
nobody      raphael    root
```

PRIMARY PROCESS Command

The PRIMARY PROCESS command causes the manager process to switch to its backup processor. The primary processor then becomes the backup processor.

The PRIMARY PROCESS command has the following syntax:

```
PRIMARY PROCESS process-name , backup-processor-number
```

`PROCESS` *`process-name`*

specifies the name of the PROCESS object that is to switch processors.

`backup-processor-number`

specifies the number of the backup processor.

Considerations

Consider the following points when using the PRIMARY PROCESS command:

- PRIMARY PROCESS is a sensitive command requiring super-group access.
- For this command to succeed, the backup process must be active, and the specified processor number must be the processor on which the backup process is running.
- Use the INFO PROCESS command to determine which processor is being used by the backup process.

Example

The following command switches the manager process $ZNFS to processor number 3:

```
->PRIMARY PROCESS $ZNFS, 3
```

START LAN Command

The START LAN command starts one or more LAN interface processes. A LAN interface process provides a connection from the OSS NFS subsystem to the specified TCP/IP process (for conventional NonStop TCP/IP) or to the specified TCPSAM process (for Parallel Library TCP/IP). This connection makes the OSS NFS subsystem available to clients through the specified TCP/IP interface.

For conventional NonStop TCP/IP, you can define more than one LAN object, but only one LAN interface process for each TCP/IP process can be in a STARTED state at the same time. For Parallel Library TCP/IP, you cannot define more than one LAN object for the entire Parallel Library TCP/IP subsystem. Typically, SERVER objects are started before LAN objects so that remote clients do not see a partial file hierarchy.

The START LAN command has the following syntax:

```
START LAN lan-name [ , SEL [ NOT ] state ]
```

`LAN lan-name`

specifies the name of the LAN object to be started.

`SEL [ NOT ] state`

specifies the summary state select option. You can use the SEL option to specify that you want to start only LAN objects in the STOPPED summary state.

Considerations

Consider the following points when using the START LAN command:

- START LAN is a sensitive command requiring super-group access.
- Before a LAN object is started the following requirements must be met:
  - The LAN object must be defined by using an ADD LAN command.
  - The RPC port mapper process, the TCP/IP process, and the TCP/IP subnetwork object controlling the TCP/IP line must exist and be in the STARTED summary state.
  - The LAN object must be in the STOPPED summary state.
- Successfully starting a LAN object causes the SUBSYS object to enter the STARTED summary state.
- The LAN interface process does not run as a true process pair, but runs as a single process with a persistent backup. If the primary processor in which the LAN interface process is to run fails before the command is executed (and the specified backup processor is up and running), the manager process automatically attempts to restart the LAN interface process in the backup processor.
- You can also start some LAN objects using the START SUBSYS command. START SUBSYS starts all restartable LAN interface and server processes. You can determine whether a LAN interface process is restartable by using the STATUS LAN command.
- If during initialization, the LAN interface process is unable to communicate with the appropriate NonStop TCP/IP subnetwork object (the one that controls the TCP/IP line to be used by the LAN object), the LAN interface process terminates, indicating a fatal error, and the LAN object is returned to the STOPPED summary state. It is the responsibility of the operator to restart the LAN object when the problem is corrected.
- To terminate the operation of a LAN object, use the STOP LAN or ABORT LAN command.

Example

The following command starts a LAN object named LAN0:

```
->START LAN $ZNFS.LAN0
```

START SERVER Command

The START SERVER command starts one or more server processes. Starting a server has the effect of performing a local mount using the server's mount point. You must start SERVER objects in the correct order; that is, start at the root of the local file hierarchy so that each server's mount point is defined at the time it is started.

You can define any number of SERVER objects, but only the number of servers specified in the MAXSERVERS startup parameter can be in a STARTED state at the same time. (If no MAXSERVERS value is specified, only 16 server processes can be in a STARTED state at the same time.)

The START SERVER command has the following syntax:

```
START SERVER server-name [ , SEL [ NOT ] state ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object to be started.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to start only SERVER objects in the STOPPED summary state.

Considerations

Consider the following points when using the START SERVER command:

- START SERVER is a sensitive command requiring super-group access or super-ID access if NULL-ALIAS-OK and ROOT-USER-OK are TRUE.
- Before you start a SERVER object, the SERVER object must be defined by using an ADD SERVER command and must be in the STOPPED summary state.
- Successfully starting a SERVER object causes the SUBSYS object to enter the STARTED summary state.
- If you use an object-name template to start multiple SERVER objects, the SERVER objects are started in alphabetical order by mount point.
- You can also start some SERVER objects using the START SUBSYS command. START SUBSYS starts all restartable LAN interface and server processes in the same sequence that they were originally started. You can determine whether a server process is restartable by using the STATUS SERVER command.

- It is recommended that server processes be run in persistent mode. If the primary processor in which a server process is to run fails, the START SERVER command automatically attempts to start the server process in the backup processor.
- To terminate the operation of a SERVER object, use the STOP SERVER or ABORT SERVER command.

Example

The following command starts a server named NSERV11:

```
->START SERVER $ZNFS.NSERV11
```

START SUBSYS Command

The START SUBSYS command cancels the effect of the STOP SUBSYS command and starts all restartable LAN interface and server processes that were in the STARTED summary state when the STOP SUBSYS command was executed. You can determine whether a LAN interface or server process is restartable by using the STATUS LAN or STATUS SERVER command.

Although there is no reason a START SUBSYS command could not be issued to initiate a brand new subsystem, the command serves no real purpose in this situation because starting any LAN or SERVER object automatically causes the SUBSYS object to enter the STARTED state.

The START SUBSYS command has the following syntax:

```
START SUBSYS process-name

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

      [ , SEL [ NOT ] state ]
```

`SUBSYS` *`process-name`*

specifies the name of the SUBSYS object to be started. The SUBSYS object uses the name of the manager process with which it is associated.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can use the SUB option to specify that you want to start only specified subordinate objects. See Examples on page 2-70 for an illustration of the effect of SUB ALL.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to start only objects in the STOPPED summary state.

Considerations

Consider the following points when using the START SUBSYS command:

- START SUBSYS is a sensitive command requiring super-group access.
- When you start the SUBSYS object, the SUBSYS object must be in the STOPPED summary state.
- The SUBSYS object is considered to be in the STARTED summary state upon the successful completion of a START SUBSYS command or upon the successful starting of any LAN interface or server process. Once started, the SUBSYS object can only be stopped by issuing a STOP or ABORT SUBSYS command or by externally stopping the manager process; explicitly stopping all LAN interface and server processes does not stop the SUBSYS object.
- Starting a server has the effect of performing a local mount using the server's mount point. You must start SERVER objects in the correct order; that is, start at the root of the local file hierarchy so that each server's mount point is defined at the time it is started.

  If you use the START SUBSYS command to start multiple SERVER objects, the SERVER objects are started in the same sequence that they were originally started.
- When used with a SUBSYS object, the SUB and SEL options operate just as they do with any other objects. However, when you specify SUB ALL for a SUBSYS object and specify a state with the SEL option, the set of objects selected is affected by changes in the states of PROCESS, LAN, and SERVER objects caused by the operation. For further details, see the following examples.

Examples

Assume LAN0 and SRV1 are restartable processes, and SRV2 and SRV3 are not restartable.

The following command starts the OSS NFS subsystem associated with the manager process $ZNFS, and any restartable LAN and SERVER objects:

```
->START SUBSYS $ZNFS
```

This command would start LAN 1 and SRV1, but not SRV2 or SRV3.

The next command starts the OSS NFS subsystem, any restartable LAN and SERVER objects, and all LAN and SERVER objects in the STOPPED summary state:

```
->START SUBSYS, SUB ALL, SEL STOPPED
```

Again, the LAN0 and SRV1 objects are started as part of starting the subsystem because they are restartable processes. The SRV2 and SRV3 objects, although not restartable, are started because they are subordinate objects in the STOPPED summary state (that is, when the SUB option is applied to a START SUBSYS command, you are actually applying the START command to a different set of objects and the command then executes according to the rules for that particular object. In this case, the SUB ALL option, results in the generation of a START SERVER command.)

You should not include the SUB ALL option without the SEL STOPPED option, or the following operations occur:

- LAN0 and SRV1 are started as part of starting the subsystem.
- Applying the START command to all subordinate LAN objects generates a warning that LAN0 is already started.
- Applying the START command to all subordinate SERVER objects generates a warning that SRV1 is already started.
- An error is generated because the START command is applied to subordinate objects that are not valid with the START command: for example, GROUP objects.

STATS LAN Command

The STATS LAN command returns statistics for a LAN object.

Whenever you include the RESET option, the counters associated with the specified objects are displayed; then they are reset to 0 and the timestamp for the reset is recorded. Any STATS LAN command returns the time at which the current statistics were sampled and the time at which the counters were last reset.

The information contained in the displays produced by STATS LAN is described in detail under Example on page 2-72.

The STATS LAN command has the following syntax:

```
STATS LAN lan-name [ , RESET ] [ , SEL [ NOT ] state ]
```

`LAN lan-name`

specifies the name of the LAN object for which statistics information is requested.

`\RESET`

specifies that the statistics counters should be reset to 0 after the statistics are retrieved.

`SEL [ NOT ] state`

specifies the summary state select option. You can use the SEL option to specify that you only want statistics returned for LAN objects in the STARTED summary state.

Considerations

Consider the following points when using the STATS LAN command:

- STATS LAN is a nonsensitive command without the RESET option, but a sensitive command with the RESET option requiring super-group access.
- When you issue a STATS LAN command, the LAN object must be in the STARTED summary state.

Example

The following command displays statistics for the LAN object named LAN:

```
->STATS LAN $ZNFS.LAN
```

The resulting display has the following format:

```
NFS Stats LAN \SYS1.$ZNFS.LAN

Reset Time.... 28 Aug 1995, 11:33:58.632
Sample Time... 28 Aug 1995, 12:09:24.328

Bad Calls  .......... 0
I/O Errors .......... 0
Messages Dropped..... 0

Message Sizes
      <=128  <=256  <=512  <=1024  <=2048   <=4096   >4096
Recv     14      0     48       0       0        0       0
Send      0     24     42      48       0        0       0

Procedure              Name      Count   Elapsed Secs    Avg. Resp
   100003 (2).  4  NFS LOOKUP       21         0 .461      0 .021
   100003 (2).  6  NFS READ         10         0 .494      0 .049
   100003 (2). 16  NFS READDIR       6         0 .120      0 .020
   100003 (2). 17  NFS STATFS        2         0 .232      0 .116
   100005 (1).  1  MNT MNT           2         0 .608      0 .304
   100005 (1).  3  MNT UMNT          3         0 .366      0 .122
```

The display contains the following information:

Reset Time	is the time the statistics were last reset using the RESET option. If the statistics have never been reset, the reset time is the time the LAN interface process was started.
Sample Time	is the time at which the statistics sample was taken.
Bad Calls	is the number of RPC requests rejected without being processed. This typically occurs due to errors in decoding the RPC message contents.
I/O Errors	is the number of RPC requests rejected because of I/O errors in communicating with the manager process or a server process.
Messages Dropped	is the number of messages that have been dropped because of overflows of internal queues or the receipt of duplicate requests.
Message Sizes	is a histogram of the number of messages of specified lengths received and sent by the LAN interface process. You can tune the buffer size in the LAN interface, based on typical message lengths.
Procedure	is the number of a procedure called by the LAN interface process in the form *`RPC-program (version).procedure`* or *`MNT-program (version .procedure`*. The procedures conform to the NFS Remote Procedure Call (RPC) and Mount protocols.

Name	is the name of the procedure.
Count	is the number of times the procedure has been called by the LAN interface process. Procedures with zero counts are not displayed.
Elapsed Secs	is the total elapsed (wall-clock) time required for the LAN interface process to reply to an RPC call.
Avg. Resp	is the average response time in seconds, which is calculated by dividing the elapsed time by the count.

STATS PROCESS Command

The STATS PROCESS command returns statistics for all objects that meet the specified *process-name*, SUB, and SEL specifications.

Whenever you include the RESET option, the counters associated with the specified objects are displayed; then they are reset to 0 and the timestamp for the reset is recorded. Any STATS PROCESS command returns the time at which the current statistics were sampled and the time at which the counters were last reset.

The information contained in the displays produced by STATS PROCESS is described in detail under Example on page 2-74.

The STATS PROCESS command has the following syntax:

```
STATS PROCESS process-name [ , RESET ]

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

      [ , SEL [ NOT ] state ]
```

`PROCESS` *`process-name`*

specifies the name of the PROCESS object for which statistics information is requested.

`RESET`

specifies that the statistics counters should be reset to 0 after the statistics are retrieved.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can specify SUB ALL to request statistics about all objects subordinate to the PROCESS object, or you can specify one of the object types to request information about that object.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you only want statistics returned for objects in the STARTED summary state.

Considerations

Consider the following points when using the STATS PROCESS command:

- STATS PROCESS is a nonsensitive command without the RESET option, but a sensitive command with the RESET option requiring super-group access.
- The information contained in the displays produced for subordinate objects is described under the individual command descriptions for each object.

Example

The following command displays statistics for the manager process, and then resets the counters to zero:

```
->STATS PROCESS $ZNFS, RESET
```

The resulting display has the following format:

```
NFS Stats PROCESS \SYS1.$ZNFS

Extended Memory Pool
  Reset Time.......... 28 Aug 1995, 11:33:13.259
  Sample Time......... 28 Aug 1995, 12:10:50.890
  Configured Size .... 96256
  Current Size........ 89484    Maximum Used........ 95560
  Current Fragments... 4        Maximum Fragments... 17
  Growth Count........ 46       Fail Count.......... 0
  Growth Time......... 28 Aug 1995, 11:50:03.285

Receive Queue
  Reset Time.......... 28 Aug 1995, 11:33:13.259
  Sample Time......... 28 Aug 1995, 12:10:50.890
  Configured Limit.... 2000
  Current Usage....... 1        Maximum Used........ 2

Openers
  Reset Time.......... 28 Aug 1995, 11:33:13.259
  Sample Time......... 28 Aug 1995, 12:10:50.890
  Configured Limit.... 33
  Current Usage....... 2        Maximum Used........ 2
```

The display contains the following information:

Extended Memory Pool	is the heading for statistics about the use of extended memory in the manager process. When you specify RESET, only the Maximum Used, Fail Count, and Reset Time values are reset.
Reset Time	is the time the PROCESS statistics were last reset using the RESET option. If the PROCESS statistics have never been reset, the reset time is the time the manager process was started.

Sample Time	is the time at which the PROCESS statistics sample was taken.
Configured Size	is the total number of bytes configured for the extended memory pool.
Current Size	is the number of bytes currently allocated from the extended memory pool. If the number of bytes allocated are consistently near or over one-half the total pool size, the pool might be overused.
Maximum Used	is the maximum number of bytes that have ever been allocated by the extended memory pool at any one time since the statistics were reset, the process was initiated, or a backup takeover occurred.
Current Fragments	is the number of fragments available in storage.
Maximum Fragments	is the maximum number of fragments in available storage since the PROCESS statistics were reset, the process was initiated, or a backup takeover occurred.
Growth Count	is the number of times the extended memory pool was enlarged due to storage pressure.
Fail Count	is the amount of storage (in bytes) requested but rejected since the PROCESS statistics were reset, the process was initiated, or a backup takeover occurred.
Growth Time	is the time when the extended memory pool was last enlarged.
Receive Queue	is the heading for statistics about unanswered requests on the manager process receive queue.
Reset Time	is the time the PROCESS statistics were last reset using the RESET option. If the PROCESS statistics have never been reset, the reset time is the time the manager process was started.
Sample Time	is the time at which the PROCESS statistics sample was taken.
Configured Limit	is the upper limit of requests in the receive queue.
Current Usage	is the current number of requesters in the receive queue.
Maximum Used	is the maximum number of requesters in the receive queue.
Openers	is the heading for statistics about the number of process openers.
Reset Time	is the time the PROCESS statistics were last reset using the RESET option. If the PROCESS statistics have never been reset, the reset time is the time the manager process was started.

Sample Time	is the time at which the PROCESS statistics sample was taken.
Configured Limit	is the maximum number of openers allowed.
Current Usage	is the current number of openers.
Maximum Used	is the maximum number of openers since the statistics were reset, the process was initiated, or a backup takeover occurred.

STATS SERVER Command

The STATS SERVER command returns statistics for a SERVER object.

Whenever you include the RESET option, the counters associated with the specified objects are displayed; then they are reset to 0 and the timestamp for the reset is recorded. Any STATS SERVER command returns the time at which the current statistics were sampled and the time at which the counters were last reset.

The information contained in the displays produced by STATS SERVER is described in detail under Example on page 2-77.

The STATS SERVER command has the following syntax:

```
STATS SERVER server-name [ , RESET ] [ , SEL [ NOT ] state ]
```

`SERVER server-name`

specifies the name of the SERVER object for which statistics information is requested.

`RESET`

specifies that the statistics counters should be reset to 0 after the statistics are retrieved.

`SEL [ NOT ] state`

specifies the summary state select option. You can use the SEL option to specify that you only want statistics returned for SERVER objects in the STARTED summary state.

Considerations

Consider the following points when using the STATS SERVER command:

- STATS SERVER is a nonsensitive command without the RESET option, but a sensitive command with the RESET option requiring super-group access.
- When you issue a STATS SERVER command, the SERVER object must be in the STARTED summary state.

Example

The following command displays statistics for the server named $ZNFS.ROOT:

```
->STATS SERVER $ZNFS.ROOT
```

The resulting display has the following format:

```
NFS Stats SERVER \KT22.$ZNFS.ROOT

Reset Time.... 21 Sep 1995, 18:57:16.601
Sample Time... 21 Sep 1995, 18:57:30.272

Bad Calls........... 0
I/O Errors.......... 0
Messages Dropped..... 0

Cache Type  Maximum  Current  Misses  Read Hits Write Hits
  Trans          32        32       0           0
```

The display contains the following information:

Reset Time	is the time the statistics were last reset using the RESET option. If the statistics have never been reset, the reset time is the time the server process was started.
Sample Time	is the time at which the statistics sample was taken.
Bad Calls	is the number of RPC requests rejected without being processed. This typically occurs due to errors in decoding the RPC message contents.
I/O Errors	is the number of RPC requests rejected because of I/O errors in communicating with the LAN interface process or a disk process.
Messages Dropped	is the number of messages that have been dropped because of overflows of internal queues or the receipt of duplicate requests.
Cache Type	is the heading that identifies the type of cache to which the following statistics apply.
Trans	is the heading for the transaction cache statistics. The transaction cache is used by the server to keep track of recent responses to nonidempotent NFS operations. Idempotent operations are operations that can be performed again and again with the same response. For the transaction cache, the statistics displayed can be interpreted as follows:

Maximum	indicates the maximum number of saved transactions.
Current	indicates the current number of saved transactions.
Misses	indicates the number of requests requiring new operations.

Read Hits	indicates the number of times the server did not have to reperform an operation because it could return the previous response from a duplicate request.
Write Hits	is not applicable for the transaction cache.

STATUS LAN Command

The STATUS LAN command reports the status of a LAN object. You can display status information about LAN objects with or without detail.

The STATUS LAN command has the following syntax:

```
STATUS LAN lan-name [ , DETAIL ] [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object for which status information is requested.

`DETAIL`

specifies that detailed LAN status information is requested.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you are requesting status information about only LAN objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the STATUS LAN command:

- STATUS LAN is a nonsensitive command.
- The summary state of an object does not prevent the STATUS LAN command from being completed successfully.
- The STATUS LAN command does not alter the summary state of objects.

Examples

The following command reports the status of all LAN objects:

```
->STATUS LAN $ZNFS.*
```

The resulting display has the following format:

```
NFS Status LAN \SYS1.$ZNFS.NFSLAN

Objname  State     Restart  System   PPID    Trace    LastErr
NFSLAN   STARTED      YES      251     1,33      OFF         0
```

The display contains the following information:

Objname	is the name of the LAN object.
State	is the summary state of the LAN object.
Restart	indicates whether a subsequent START SUBSYS command will attempt to restart the LAN interface process (YES) or will not attempt to restart it (NO). The Restart attribute is turned on when the LAN object is placed in the STARTED summary state and is only turned off when the LAN interface process is stopped by an SCF STOP or ABORT command.
System	is the number of the system on which the LAN interface process is running. If the LAN object is not started, this field is blank.
PPID	is the processor number and process identification number (PIN) of the LAN interface process. If the LAN object is not started, this field is blank.
Trace	indicates whether the LAN interface process is being traced (ON) or not (OFF).
LastErr	is the number of the last error encountered by the LAN interface process.

The following command displays the detailed status information for a LAN object named $ZNFS.NFSLAN:

```
->STATUS LAN $ZNFS.NFSLAN, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Status LAN \SYS1.$ZNFS.NFSLAN

State............. STARTED       System Name....... \KT22
System Number..... 212           Primary PID....... 0,72
Restart........... YES           Trace............. ON
Trace File........ \SYS1.$SYSTEM.ZOSSNFS.TEMP
Last Error........ 26
```

The detailed display contains the following information:

State	is the summary state of the LAN object.
System Name	is the name of the system on which the LAN interface process is running.
System Number	is the number of the system on which the LAN interface process is running. If the LAN object is not started, this field is blank.

Primary PID	indicates the processor number and process identification number (PIN) of the LAN interface process.
Restart	indicates whether a subsequent START SUBSYS command will attempt to restart the LAN interface process (YES) or will not attempt to restart it (NO). The Restart attribute is turned on when the LAN object is placed in the STARTED summary state and is only turned off when the LAN interface process is stopped by an SCF STOP LAN or ABORT LAN command.
Trace	indicates whether the LAN interface process is being traced (ON) or not (OFF).
Trace File	indicates the name of the file receiving the trace output.
Last Error	is the number of the last error encountered by the LAN interface process.

STATUS PROCESS Command

The STATUS PROCESS command returns status information for all objects that meet the specified *process-name*, SUB, and SEL specifications.

You can display status information about the specified objects with or without detail.

The information contained in the displays produced by STATUS PROCESS is described in detail under Examples on page 2-81.

The STATUS PROCESS command has the following syntax:

```
STATUS PROCESS process-name [ , DETAIL ]

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

       [ , SEL [ NOT ] state ]
```

`PROCESS` *`process-name`*

specifies the name of the PROCESS object for which status information is requested.

`DETAIL`

specifies that detailed status information is requested.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can specify SUB ALL to status information about the PROCESS object and all objects subordinate to the PROCESS object, SUB ONLY to request information about all objects subordinate to the PROCESS object, or specify one of the object types to request information about that object.

`SEL [ NOT ] `*`state`*

specifies the summary state select option. You can use the SEL option to specify that you are requesting status information about only objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the STATUS PROCESS command:

- STATUS PROCESS is a nonsensitive command.
- The STATUS PROCESS command does not alter the summary state of objects.
- The information contained in the displays produced for subordinate objects is described under the individual command descriptions for each object.

Examples

The following command reports the status of the $ZNFS process:

```
->STATUS PROCESS $ZNFS
```

The resulting display has the following format:

```
NFS Status PROCESS \SYS1.$ZNFS

Objname  State   Tkovrs  Backup  System  PPID  BPID  Trace LastErr
$ZNFS    STARTED    0   STOPPED   206    1,194        OFF     0
```

The display contains the following information:

Objname	is the name of the PROCESS object.
State	is the summary state of the primary process.
Tkovrs	is the number of takeovers by the backup process that have occurred.
Backup	is the state of the backup process.
System	is the number of the system on which the process is running.
PPID	is the process ID of the primary process.
BPID	is the process ID of the backup process. If the backup process is not running, this field is blank.
Trace	indicates whether the process is being traced (ON) or not (OFF).
LastErr	is the number of the last error encountered.

The following command displays the detailed status information for the $ZNFS process:

```
->STATUS PROCESS $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Status PROCESS \SYS1.$ZNFS

System Name....... \SYS1          System Number..... 212
State............. STARTED        Primary PID....... 1,59
Backup State...... STOPPED        Backup PID........ NONE
Trace............. OFF            Takeovers......... 0
Last Error........ 0
```

The detailed display contains the following information:

System Name	is the name of the system on which the process is running.
System Number	is the number of the system on which the process is running.
State	is the summary state of the primary process.
Primary PID	is the process ID of the primary process.
Backup State	is the state of the backup process.
Backup PID	is the process ID of the backup process. If the backup process is not running, the value NONE is displayed.
Trace	indicates whether the process is being traced (ON) or not (OFF).
Takeovers	is the number of takeovers by the backup process that have occurred.
Last Error	is the number of the last error encountered.

STATUS SERVER Command

The STATUS SERVER command reports the status of a SERVER object.

You can display status information about SERVER objects with or without detail.

The information contained in the displays produced by STATUS SERVER is described in detail under Examples on page 2-83.

The STATUS SERVER command has the following syntax:

```
STATUS SERVER server-name [ , DETAIL ] [ , SEL [ NOT ] state
]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object for which status information is requested.

`DETAIL`

specifies that detailed SERVER status information is requested.

`SEL [ NOT ] state`

specifies the summary state select option. You can use the SEL option to specify that you are requesting status information about only SERVER objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the STATUS SERVER command:

- STATUS SERVER is a nonsensitive command.
- The summary state of an object does not prevent the STATUS SERVER command from being completed successfully.
- The STATUS SERVER command does not alter the summary state of objects.

Examples

The following command reports the status of all servers:

```
->STATUS SERVER $ZNFS.*
```

The resulting display has the following format:

```
NFS Status SERVER \SYS1.$ZNFS.*

Objname  State    Restart  System  PPID   BPID   Trace  LastErr
SRV1     STARTED    YES     251    0,43          OFF      3
PSRV2    STOPPED    NO                           OFF      0
NSRV3     STARTED   YES     251    0,44   1,37   OFF      0
```

The display contains the following information:

Objname	is the name of the SERVER object.
State	is the summary state of the SERVER object.
Restart	indicates whether a subsequent START SUBSYS command will attempt to restart the server process (YES) or will not attempt to restart it (NO). The Restart attribute is turned on when the SERVER object is placed in the STARTED summary state and is only turned off when the server process is stopped by an SCF STOP SERVER or ABORT SERVER command.
System	is the number of the system on which the server process is running. If the SERVER object is not started, this field is blank.
PPID	is the processor number and process identification number (PIN) of the server process. If the SERVER object is not started, this field is blank.
BPID	is the processor number and PIN of the backup server process. If the backup server process is not started, this field is blank.
Trace	indicates whether the server process is being traced (ON) or not (OFF).

LastErr	is the number of the last error encountered by the server process.

The following command displays the detailed status information for a SERVER object named $ZNFS.ROOT:

```
->STATUS SERVER $ZNFS.ROOT, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Status SERVER \SYS1.$ZNFS.ROOT

State.............STARTED        Restart........... YES
System Name.......\SYS1          System Number..... 206
Primary PID.......6,62           Backup PID........ NONE
Last Error........3              Trace............. ON
Last Error Detail NFS E000003 NFS2 LOOKUP error 2 on /foo No such file
Trace Filename.... $DATA.NFSTRACE.SRV1LOG
```

The detailed display contains the following information:

State	is the summary state of the SERVER object.
Restart	indicates whether a subsequent START SUBSYS command will attempt to restart the server process (YES) or will not attempt to restart it (NO). The Restart attribute is turned on when the SERVER object is placed in the STARTED summary state and is only turned off when the server process is stopped by an SCF STOP SERVER or ABORT SERVER command.
System Name	is the name of the system on which the server process is running.
System Number	is the number of the system on which the server process is running. If the SERVER object is not started, the value displayed is NONE.
Primary PID	is the processor number and process identification number (PIN) of the server process. If the SERVER object is not started, this field is blank.
Backup PID	is the processor number and PIN of the backup server process. If the backup server process is not started, the value NONE is displayed.
Last Error	is the number of the last error encountered by the server process.
Trace	indicates whether the server process is being traced (ON) or not (OFF).
Last Error Detail	indicates the actual last error that occurred.
Trace Filename	indicates the name of the file receiving the trace output.

STATUS SUBSYS Command

The STATUS SUBSYS command returns status information for all objects that meet the specified *process-name*, SUB, and SEL specifications.

You can display status information about the specified objects with or without detail.

The information contained in the displays produced by STATUS SUBSYS is described in detail under Examples on page 2-86.

The STATUS SUBSYS command has the following syntax:

```
STATUS SUBSYS process-name [ , DETAIL ]

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

       [ , SEL [ NOT ] state ]
```

SUBSYS *process-name*

specifies the name of the SUBSYS object for which status information is requested. The SUBSYS object uses the name of the manager process with which it is associated.

DETAIL

specifies that detailed status information is requested.

SUB [ALL | NONE | ONLY | *subtype*]

specifies the subordinate object select option. You can specify SUB ALL to see status information about the SUBSYS object and all objects subordinate to the SUBSYS object, SUB ONLY to request information about all objects subordinate to the SUBSYS object, or specify one of the object types to request information about that object. If SUB is specified without an option, ALL is assumed.

SEL [NOT] *state*

specifies the summary state select option. You can use the SEL option to specify that you are requesting status information about only objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the STATUS SUBSYS command:

- STATUS SUBSYS is a nonsensitive command.
- The summary state of an object does not prevent the STATUS SUBSYS command from being completed successfully.
- The STATUS SUBSYS command does not alter the summary state of objects.

- The information contained in the displays produced for subordinate objects are described under the individual command descriptions for each object.

Examples

The following command reports the status of the SUBSYS object associated with the manager process $ZNFS:

```
->STATUS SUBSYS $ZNFS
```

The resulting display has the following format:

```
NFS Status SUBSYS \SYS1.$ZNFS

Objname  State   Opens  Opens Allowed   Reuse Srvr ID  LastErr
$ZNFS    STOPPED    0         YES             ON             0
```

The display contains the following information:

Objname	is the name of the manager process.
State	is the summary state of the subsystem.
Opens	is the number of opens issued against the subsystem.
Opens Allowed	indicates whether opens can be issued; that is, whether remote mount requests are accepted.
Reuse Srvr ID	indicates whether the subsystem can reuse a server ID when restarting a stopped server. This attribute can be reset by using the ALTER SUBSYS command.
LastErr	is the number of the last error encountered.

The following command displays the detailed status information for the SUBSYS object associated with the manager process $ZNFS:

```
->STATUS SUBSYS $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
NFS Detailed Status SUBSYS \SYS1.$ZNFS

State............. STOPPED         Remote Mounts..... 0
Allow Opens....... YES             Reuse Server ID... ON
Last Error........ 0
Last Error Detail. 0
```

The detailed display contains the following information:

State	is the summary state of the subsystem.
Remote Mounts	is the number of opens issued against the subsystem..
Allow Opens	indicates whether opens can be issued; that is, whether remote mount requests are accepted.

Reuse Server ID	indicates whether the subsystem can reuse a server ID when restarting a stopped server. This attribute can be reset by using the ALTER SUBSYS command.
Last Error	is the number of the last error encountered.
Last Error Detail	indicates the actual last error that occurred. This field is only present when Last Error has a nonzero value.

STOP LAN Command

The STOP LAN command terminates the operation of a LAN object in a controlled manner. The STOP LAN command does not abruptly terminate in-progress activities. When the operation is complete, the LAN object is in the STOPPED summary state.

The STOP LAN command has the following syntax:

```
STOP LAN lan-name [ , ORDERLY | , FORCED ]

     [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object whose operation is to be terminated.

`ORDERLY`

specifies that the STOP operation rejects all new requests, and waits for all active links to be dropped before terminating the object. If you omit the ORDERLY or FORCED option, the STOP operation succeeds only if no active links exist when you issue the command.

`FORCED`

specifies that the STOP operation rejects all new requests, and aborts all active links before terminating the object. If you omit the ORDERLY or FORCED option, the STOP operation succeeds only if no active links exist when you issue the command.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to abort only LAN objects in the STARTED summary state.

Considerations

Consider the following points when using the STOP LAN command:

- STOP LAN is a sensitive command requiring super-group access.
- When you stop a LAN object, it must be in the STARTED summary state.
- To stop a LAN object immediately, despite only active links, use the ABORT LAN command.

- The STOP LAN command clears the RESTART attribute. The RESTART attribute determines whether or not a subsequent START SUBSYS command initiates an attempt to restart the stopped LAN object. The STOP LAN command should only be used in situations where you will not want to restart the same configuration of LAN interface and server processes that are currently running. To stop a LAN object without clearing the RESTART attribute, use the STOP SUBSYS command.
- For LAN objects, active links are active requests that are awaiting a server response.
    - A normal stop succeeds only if there are no active requests.
    - An orderly stop causes the LAN interface process to enter the STOPPING summary state, reject new requests, and wait for all active requests to complete before stopping the LAN interface process.
    - A forced stop causes the LAN interface process to enter the STOPPING summary state, reject new requests, and abort all active links before stopping the LAN interface process.
- While a LAN object is stopped, no corresponding LAN interface process exists, and therefore no Remote Procedure Call (RPC) calls can be accepted over the TCP/IP port serviced by that process.
- Stopping a LAN object does not cause any SERVER objects to be stopped. However, when no LAN object is in the STARTED summary state, SERVER objects in the STARTED summary state are unable to receive or respond to RPC calls.
- Use the START LAN command to reinitiate the operation of the stopped LAN object.
- Use the STATUS LAN command to determine the current summary state of LAN objects.
- Use the INFO LAN command to determine the current attribute values of LAN objects.
- Use the DELETE LAN command to remove a LAN object from the OSS NFS subsystem.

Example

The following command terminates, in an orderly manner, the operation of a LAN object named NFSLAN:

```
->STOP LAN $ZNFS.NFSLAN, ORDERLY
```

STOP SERVER Command

The STOP SERVER command terminates the operation of a SERVER object in a controlled manner. The STOP SERVER command does not abruptly terminate in-progress activities. When the operation is complete, the SERVER object is in the STOPPED summary state.

The STOP SERVER command has the following syntax:

```
STOP SERVER server-name [ , ORDERLY | , FORCED ]

     [ , SEL [ NOT ] state ]
```

`SERVER` *`server-name`*

specifies the name of the SERVER object whose operation is to be terminated.

`ORDERLY`

specifies that the STOP operation rejects all new requests, and waits for all active links to be dropped before terminating the object. If you omit the ORDERLY or FORCED option, the STOP operation succeeds only if no active NFS requests exist when you issue the command.

`FORCED`

specifies that the STOP operation rejects all new requests, and aborts all active NFS requests before terminating the object. If you omit the ORDERLY or FORCED option, the STOP operation succeeds only if no active NFS requests exist when you issue the command.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to abort only SERVER objects in the STARTED summary state.

Considerations

Consider the following points when using the STOP SERVER command:

- STOP SERVER is a sensitive command requiring super-group access or super-ID access if NULL-ALIAS-OK and ROOT-USER-OK are TRUE.
- When you stop a SERVER object, it must be in the STARTED summary state.
- To stop a SERVER object immediately, despite active NFS requests, use the ABORT SERVER command.
- The STOP SERVER command clears the RESTART attribute. The RESTART attribute determines whether or not a subsequent START SUBSYS command initiates an attempt to restart the stopped SERVER object. The STOP SERVER command should only be used in situations where you will not want to restart the same configuration of LAN interface and server processes that are currently

running. To stop a SERVER object without clearing the RESTART attribute, use the STOP SUBSYS command.

- For SERVER objects, active NFS requests are requests in progress.
    - A normal stop succeeds only if there are no active requests.
    - An orderly stop causes the server to enter the STOPPING summary state, reject new requests, and wait for all active requests to complete before stopping the server process.
    - A forced stop causes the server to enter the STOPPING summary state, reject new requests, and aborts all active requests before stopping the server process.
- While a SERVER object is stopped, no corresponding server process exists, and therefore no Remote Procedure Call (RPC) calls, which deal with the directories and files serviced by that process, can be accepted.
- Use the START SERVER command to reinitiate the operation of the stopped server.
- Use the STATUS SERVER command to determine the current summary state of the SERVER objects.
- Use the INFO SERVER command to determine the current attribute values of SERVER objects.
- Use the DELETE SERVER command to remove a SERVER object from the OSS NFS subsystem.

Example

The following command terminates the operation of a server named $ZNFS.SRV1:

```
->STOP SERVER $ZNFS.SRV1
```

STOP SUBSYS Command

The STOP SUBSYS command terminates the operation of the OSS NFS subsystem in a controlled manner. The STOP SUBSYS command does not abruptly terminate in-progress activities.

Stopping the SUBSYS object causes all SERVER objects (all server processes), all LAN objects (LAN interface processes), and the PROCESS object to be placed in the STOPPED summary state.

The STOP SUBSYS command has the following syntax:

```
STOP SUBSYS process-name [ , ORDERLY | , FORCED ]

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

     [ , SEL [ NOT ] state ]
```

`SUBSYS` *`process-name`*

specifies the name of the SUBSYS object whose operation is to be terminated. The SUBSYS object uses the name of the manager process with which it is associated.

`ORDERLY`

specifies that the STOP operation rejects new requests, and waits for all active links to be dropped before terminating the object. If you omit ORDERLY, the STOP operation succeeds only if no active links exist when you issue the command.

`FORCED`

specifies that the STOP operation rejects all new requests, and aborts all active links before terminating the object. If you omit the ORDERLY or FORCED option, the STOP operation succeeds only if no active links exist when you issue the command.

`SUB [ ALL | NONE | ONLY |` *`subtype`* `]`

specifies the subordinate object select option. You can use the SUB option to specify that you want to stop only specified subordinate objects. The command STOP SUBSYS, SUB ALL is equivalent to STOP SUBSYS, because no further commands can be processed after the subordinate PROCESS object is stopped. With the STOP SUBSYS command, only the SUB NONE, SUB ONLY, SUB LAN, and SUB SERVER options are useful.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you want to stop only objects in the STARTED summary state.

Considerations

Consider the following points when using the STOP SUBSYS command:

- STOP SUBSYS is a sensitive command requiring super-group access.
- To stop the SUBSYS object immediately, despite active links, use the ABORT SUBSYS command.
- For the SUBSYS object, active links are remote mounts.
  - A normal stop succeeds only if there are no remote mounts against any part of the OSS NFS subsystem.
  - An orderly stop causes the subsystem to enter the STOPPING summary state, reject new mounts, and wait for all current mounts to be unmounted. After all remote mounts have been unmounted and all subsidiary LAN and SERVER objects have stopped, the manager process stops.
  - A forced stop causes the subsystem to enter the STOPPING summary state, reject new mounts, and abort all current mounts. After all remote mounts have been unmounted and all subsidiary LAN and SERVER objects have stopped, the manager process stops.

- The STOP SUBSYS command does not clear the RESTART attribute. The RESTART attribute determines whether or not a subsequent START SUBSYS command initiates an attempt to restart the stopped LAN and SERVER objects. STOP SUBSYS is useful in situations where you will want to restart the same configuration of LAN interface and server processes that was running when the command was issued.
- To reinitiate the operation of a stopped OSS NFS subsystem, the manager process must be restarted using a TACL RUN command or *Guardian* procedure call PROCESS_CREATE_. After the manager process is restarted, the subsystem can be restored to the same state it was in prior to the STOP SUBSYS command by using a START SUBSYS command.
- Use the STATUS SUBSYS command to determine the current summary state of the OSS NFS objects.
- To remove an object from the OSS NFS subsystem, use the DELETE command on each object to be deleted.

Note. You can use the LISTOPENS SUBSYS command to display remote mounts. However, because the remote mount information is not guaranteed to be accurate, OSS NFS might operate as if a host has a remote mount, when actually it no longer exists. If you cannot stop the subsystem because there are nonexistent remote mounts, use the ABORT SUBSYS command. Conversely, a host might operate as if it has a remote mount that OSS NFS no longer recognizes. Therefore, you should be aware that using the STOP SUBSYS command might disrupt clients in some situations.

Example

The following command terminates the operation of the OSS NFS subsystem associated with the manager process $ZNFS:

```
->STOP SUBSYS $ZNFS
```

STOPOPENS SUBSYS Command

The STOPOPENS SUBSYS command prevents new remote mounts from being processed.

The STOPOPENS SUBSYS command has the following syntax:

```
STOPOPENS [ SUBSYS ] process-name
```

`[ SUBSYS ]` *`process-name`*

specifies the name of the SUBSYS object which is to prohibit new remote mounts. The SUBSYS object uses the name of the manager process with which it is associated.

Considerations

Consider the following points when using the STOPOPENS command:

- STOPOPENS is a sensitive command requiring super-group access.
- Use this command to prevent the OSS NFS subsystem from being accessed by new clients when you are preparing to stop it.
- While the STOPOPENS command is in effect, remote mount requests receive no response. Clients attempting to perform remote mounts eventually time out and abort or retry their requests.
- The STOPOPENS SUBSYS command does not affect open requests from the OSS NFS subsystem or any remote mounts already in existence.
- To once again permit remote mounts, use the ALLOWOPENS command.

Example

The following command prevents new remote mounts from being processed by the OSS NFS subsystem associated with the manager process $ZNFS:

```
->STOPOPENS SUBSYS $ZNFS
```

TRACE LAN Command

The TRACE LAN command allows you to capture and store records for a LAN object. You can then display these records using the Ptrace facility. The TRACE LAN command can request the capture of data items, alter trace parameters that were set by a previous use of the command, or stop a previously requested trace operation.

The TRACE LAN command has the following syntax:

```
TRACE LAN lan-name

{ , STOP [ BACKUP ]                                      }
{ [ , BACKUP                                         ]
  [ , COUNT count                                    ]
  [ , NOCOLL                                         ]
  [ , PAGES pages                                    ]
  [ , RECSIZE size                                   ]
    , TO file-spec
  [ , WRAP                                           ]
  [ , SELECT { select-word...                   }    ]
  [          { ( select-word [ , select-word ]... )  }  ]  }
```

For a LAN object, `select-word` can be any of:

```
[ ALL ]
[ HOST "host-name" ]
[ MNT { DUMP    }  |  RPC { 100005.2  }
      { EXPORT  }  |      { 100005.5  }
      { MNT     }  |      { 100005.1  }
      { NULL    }  |      { 100005.0  }
      { UMNT    }  |      { 100005.3  }
      { UMNTALL }  |      { 100005.4  } ]
[ NFS { CREATE     }  |  RPC { 100003.9   }
      { GETATTR    }  |      { 100003.1   }
      { LINK       }  |      { 100003.12  }
      { LOOKUP     }  |      { 100003.4   }
      { MKDIR      }  |      { 100003.14  }
      { NULL       }  |      { 100003.0   }
      { READ       }  |      { 100003.6   }
      { READDIR    }  |      { 100003.16  }
      { READLINK   }  |      { 100003.5   }
      { REMOVE     }  |      { 100003.10  }
      { RENAME     }  |      { 100003.11  }
      { RMDIR      }  |      { 100003.15  }
      { ROOT       }  |      { 100003.3   }
      { SETATTR    }  |      { 100003.2   }
      { STATFS     }  |      { 100003.17  }
      { SYMLINK    }  |      { 100003.13  }
      { WRITE      }  |      { 100003.8   }
      { WRITECACHE }  |      { 100003.7   } ]
[ RPC program[(version)].procedure ]
[ RPCMSG ]
[ SERVER server-name ]
[ SPIMSG  ]
[ SYSMSG ]
[ UNKNOWN ]
```

`LAN` *`lan-name`*

specifies the name of the LAN object to which the trace operation applies.

`STOP [ BACKUP ]`

discontinues the trace currently in progress. If you specify BACKUP, only the backup process trace is stopped.

`BACKUP`

specifies that the backup process should receive the trace request. If you omit BACKUP, the primary process is assumed.

`COUNT` *`count`*

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

`NOCOLL`

indicates that the trace collector process should not be initiated.

`PAGES` *`pages`*

designates how much space, in 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 parameters 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` *`size`*

specifies the length of the data in the trace data records. *`size`* is an integer in the range of 16 through 4050, or it is equal to 0. The length of the trace header, which is 8 bytes, is not included in *`size`*. If you omit this option or specify 0, the default value of 120 bytes is used.

`TO` *`file-spec`*

specifies the name of the file into which the results of the trace operation are placed. It is a required option if STOP is not used.

`WRAP`

specifies that when the trace disk file end-of-file (EOF) is reached, trace data will wrap around to the beginning of the file and overwrite any data that is there.

```
SELECT { select-word...                          }
       { ( select-word [ , select-word ]... ) }
```

specifies criteria for selection. For the TRACE LAN command you can specify the following as *`select-word`*:

`ALL`

specifies that all messages of all types (that meet the other trace LAN criteria) are to be traced. If you omit SELECT, ALL is the default.

`HOST “host-name”`

specifies the name of a remote host whose calls are to be traced. If HOST values are specified, only calls from the hosts you specify are traced. If HOST values are not specified, any calls to the specified LAN interface process from any remote hosts (that meet the other trace LAN criteria) are traced.

`MNT`

specifies a remote procedure call in the mount protocol that is to be traced. The specified MNT procedures are traced only if they meet the other trace LAN criteria also.

`NFS`

specifies a remote procedure call in the NFS protocol that is to be traced. The specified NFS procedures are traced only if they meet the other trace LAN criteria also.

`RPC`

specifies a remote procedure call by its program number, version number, and procedure number. If you omit the version number, all versions are traced. The specified NFS procedures are only traced if they meet the other trace LAN criteria also.

`RPCMSG`

specifies that all RPC messages received and generated by the LAN interface process (that meet the other trace LAN criteria) are to be traced.

`SERVER` *`server-name`*

specifies the name of a server whose work is to be traced. If SERVER values are specified, only work directed to the servers you specify are traced. If SERVER values are not specified, any work sent to any server process (that meets the other trace LAN criteria) is traced.

`SPIMSG`

specifies that all SPI commands received and responses generated by the LAN interface process (that meet the other trace LAN criteria) are to be traced.

`SYSMSG`

specifies that all system messages received by the LAN interface process (that meet the other trace LAN criteria) are to be traced.

`UNKNOWN`

specifies that all other messages received by the LAN interface process (that meet the other trace LAN criteria) are to be traced.

Considerations

Consider the following points when using the TRACE LAN command:

- TRACE LAN is a sensitive command requiring super-group access.
- When you issue a TRACE LAN command, the LAN object must be in the STARTED summary state.
- All selection specifications must be specified in a single TRACE LAN command. Selection criteria specified in subsequent TRACE LAN commands replaces the prior specification.

- Duplicate or overlapping bit mask selection criteria are permitted, but the effect varies based on the specific values. If you specify more than one HOST, SERVER, RPC, MNT, or NFS value, or more than one value in the message category (the RPCMSG, SPIMSG, SYSMSG, UNKNOWN, or ALL values), the values are ORed. For example, if you specify ALL and SPIMSG, the effect is the same as if you only specified ALL However, when you specify values across these categories, the values are ANDed. For example, if you specify RPCMSG and NFS CREATE, only NFS CREATE messages are traced.
- In the TRACE LAN command, the HOST, MNT, NFS, and RPC specifications are ignored if RPC messages are not being traced.

Examples

The following command traces the NFS READ and NFS WRITE procedures for LAN0:

```
->TRACE LAN $ZNFS.LAN0, SELECT ( NFS READ, NFS WRITE)
```

The following command traces SPI messages generated by LAN0 for the ROOT server:

```
->TRACE LAN $ZNFS.LAN0, SELECT ( SPIMSG, SERVER ROOT )
```

VERSION LAN Command

The VERSION LAN command displays the OSS NFS product name, product number, and release date of the specified LAN object.

The VERSION LAN command has the following syntax:

```
VERSION LAN lan-name  [ , DETAIL ] [ , SEL [ NOT ] state ]
```

`LAN` *`lan-name`*

specifies the name of the LAN objects for which version information is requested.

`DETAIL`

specifies that SCF version information is also requested. For a description of the type and format of the SCF version information returned, see VERSION NULL Command on page 2-98.

`SEL [ NOT ]` *`state`*

specifies the summary state select option. You can use the SEL option to specify that you are requesting version information about only LAN objects in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the VERSION LAN command:

- VERSION LAN is a nonsensitive command.
- When you issue a VERSION LAN command, the LAN object must not be in the STOPPED summary state.

Example

The following command displays version information about the LAN object named NFSLAN:

```
->VERSION LAN $ZNFS.NFSLAN
```

The resulting display has the following format:

```
VERSION LAN \SYS1.$ZNFS.NFSLAN: NFS - T9628G00 - NFSLAN
```

VERSION NULL Command

The VERSION NULL command displays the OSS NFS product name, product number, and release date of the OSS NFS subsystem.

The VERSION NULL command has the following syntax:

```
VERSION process-name [ , DETAIL ]

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

        [ , SEL [ NOT ] state ]
```

`process-name`

specifies the name of the object for which version information is requested.

`DETAIL`

specifies that SCF version information is also requested. The DETAIL option for any VERSION command displays the following:

- The system on which the subsystem is running
- The requested subsystem version string
- The NonStop Kernel version on the system on which the subsystem is running
- The system on which SCF is running, if different
- The NonStop Kernel version for the SCF system, if the system is different
- The SCF version

- The product module version within SCF.

`SUB [ ALL | NONE | ONLY | *subtype* ]`

specifies the subordinate object select option. You can specify SUB ALL to request information about all objects subordinate to the object, or you can specify one of the object types to request information about that object.

`SEL [ NOT ] *state*`

specifies the summary state select option. You can use the SEL option to specify that you are requesting version information about objects only in a specified summary state. See “Summary States,” in Section 1, for the names and descriptions of these states.

Considerations

Consider the following points when using the VERSION NULL command:

- VERSION NULL is a nonsensitive command.

Examples

The following command displays version information about the OSS NFS subsystem that has a manager process named $ZNFS:

```
->VERSION $ZNFS
```

The resulting display has the following format:

```
VERSION PROCESS \SYS1.$ZNFS: NFS (T9628D40 01FEB96 01FEB96 )
```

The following command displays version information about a local OSS NFS subsystem that has a manager process named $ZNFS and local SCF version information:

```
->VERSION $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
Detailed VERSION \SYS1.$ZNFS
  SYSTEM  \SYS1
    NFS (NFSD40 01FEB96 01FEB96 )
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082C30 - (30NOV95) (06NOV95)
    NFS PM - NFS - T9628D40 - O1FEB96 O1FEB96
```

The following command displays version information about a remote OSS NFS subsystem that has a manager process named $ZNFS and local SCF version information:

```
->VERSION $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
Detailed VERSION \SYS5.$ZNFS
  SYSTEM  \SYS5
    NFS (NFSD40 01FEB96 01FEB96 )
    GUARDIAN - T9050 - (N40)
  SYSTEM  \SYS1
    NFS (NFSD40 01FEB96 01FEB96 )
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082C30 - (30NOV95) (06NOV95)
    NFS PM - NFS - T9628D40 - 01FEB96 01FEB96
```

VERSION PROCESS Command

The VERSION PROCESS command displays the OSS NFS product name, product number, and release date of the PROCESS object.

The VERSION PROCESS command has the following syntax:

```
VERSION PROCESS process-name [ , DETAIL ]

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

        [ , SEL [ NOT ] state ]
```

PROCESS *process-name*

specifies the name of the PROCESS object for which version information is requested.

DETAIL

specifies that SCF version information is also requested. For a description of the type and format of the SCF version information returned, see VERSION NULL Command on page 2-98.

SUB [ALL | NONE | ONLY | *subtype*]

specifies the subordinate object select option. You can specify SUB ALL to request information about all objects subordinate to the PROCESS object, or you can specify one of the object types to request information about that object.

SEL [NOT] *state*

specifies the summary state select option. You can use the SEL option to specify that you are requesting version information about objects only in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the VERSION PROCESS command:

- VERSION PROCESS is a nonsensitive command.

Examples

The following command displays the banner of the process $ZNFS:

```
->VERSION PROCESS $ZNFS
```

The resulting display has the following format:

```
VERSION PROCESS \SYS1.$ZNFS: NFS (T9628D40 01FEB96 01FEB96)
```

The following command displays the banner for the process $ZNFS and SCF version information:

```
->VERSION PROCESS $ZNFS, DETAIL
```

The resulting detailed display has the following format:

```
Detailed VERSION PROCESS \SYS1.$ZNFS
  SYSTEM \SYS1
    NFS (T9628D40 01FEB96 01FEB96)
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082D30 - (30NOV95) (06NOV95)
    NFS PM - NFS - T9628D40 - 01FEB96 01FEB96
```

VERSION SERVER Command

The VERSION SERVER command displays the OSS NFS product name, product number, and release date of the SERVER object.

The VERSION SERVER command has the following syntax:

```
VERSION SERVER process-name [ , DETAIL ]

        [ , SEL [ NOT ] state ]
```

SERVER *process-name*

specifies the name of the SERVER object for which version information is requested.

DETAIL

specifies that SCF version information is also requested. For a description of the type and format of the SCF version information returned, see VERSION NULL Command on page 2-98.

```
SEL [ NOT ] state
```

specifies the summary state select option. You can use the SEL option to specify that you are requesting version information about objects only in a specified summary state. See Summary States on page 1-7 for the names and descriptions of these states.

Considerations

Consider the following points when using the VERSION SERVER command:

- VERSION SERVER is a nonsensitive command.
- When you issue a VERSION SERVER command, the SERVER object must not be in a STOPPED state.

Examples

The following command displays information about the server named $ZNFS.NSVR3:

```
->VERSION SERVER $ZNFS.NSVR3
```

The resulting display has the following format:

```
VERSION SERVER \SYS1.$ZNFS.NSVR3: NFS - T9628D40 - 01FEB96 01FEB96
```

The following command displays detailed information the same server $ZNFS.NSVR3:

```
->VERSION SERVER $ZNFS.NSVR3, DETAIL
```

The resulting detailed display has the following format:

```
Detailed VERSION SERVER \SYS1.$ZNFS.NSVR3
  SYSTEM \SYS1
    NFS - 9628D40 - 01FEB96 01FEB96 - ZNFSSVR
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082D30 - (30NOV95) (06NOV95)
    NFS PM - NFS - T9628D40 - 01FEB96 01FEB96
```

A SCF Command Summary for OSS NFS

This appendix lists, in alphabetic order, the syntax for each of the SCF commands supported for the OSS NFS subsystem. It is included here as a quick reference for those already familiar with the SCF commands for OSS NFS.

```
ABORT LAN lan-name [ , SEL [ NOT ] state ]
```

```
ABORT SERVER server-name [ , SEL [ NOT ] state ]
```

```
ABORT SUBSYS process-name
     [ , SUB [ ALL | NONE | ONLY | subtype  ] ]
     [ , SEL [ NOT ] state ]
```

```
ADD EXPORT export-name [ [ , attribute-spec ] ... ]
```

attribute-spec for ADD EXPORT is:

```
[ ACCESS { name                     }
         { ( name [ , name ]... )  }  ]
```

```
ADD GROUP group-name [ [ , attribute-spec ]... ]
```

attribute-spec for ADD GROUP is:

```
GROUPID group-id
[ MEMBER { user-name                          }
         { ( user-name [ , user-name ]... ) } ]
```

```
ADD LAN lan-name [ [ , attribute-spec ]... ]
```

attribute-spec for ADD LAN is:

```
CPU cpu-number
PROCESS process-name
PROGRAM file-name
[ ADDR-CHECK { ON | OFF } ]
[ BACKUP backup-cpu-number ]
[ DOMAIN { domain-name                              }
         { ( domain-name [ , domain-name ]... )  } ]
[ HISTOGRAM ( message-length [ , message-length ]...) ]
[ PRI cpu-priority ]
[ SWAP file-name ]
[ TCPIP-HOST-FILE tcpip-host-file ]
[ TCPIP-PROCESS-NAME tcpip-process-name ]
[ TCPIP-RESOLVER-NAME tcpip-resolver-name ]
```

```
ADD NETGROUP netgroup-name [ [ , attribute-spec ]... ]
```

attribute-spec for ADD NETGROUP is:

```
[ netgroupx-name ]
[ ( [ host-name ] , [ user-name ] , [ domain-name ] ) ]
```

```
ADD SERVER server-name [ [ , attribute-spec ]... ]
```

attribute-spec for ADD SERVER is:

```
CPU cpu-number
MNTPOINT path-name
PROGRAM file-name
PROCESS process-name
STYPE OSS
[ BACKUP backup-cpu-number ]
[ FILE-OPT-OK { TRUE | FALSE } ]
[ MAX-FILE-SIZE integer ]
[ NULL-ALIAS-OK { TRUE | FALSE } ]
[ PRI cpu-priority ]
[ READ-ONLY { TRUE | FALSE } ]
[ ROOT-USER-OK { TRUE | FALSE } ]
[ SWAP file-name ]
[ UPSHIFT { TRUE | FALSE } ]
[ WRITE-THRU { TRUE | FALSE } ]
```

```
ADD USER user-name [ [ , attribute-spec ]... ]
```

attribute-spec for ADD USER is:

```
GROUPID group-id
USERID user-id
[ ALIAS OSS { user-name   }
            { user-number } ]
[ COMMENT “string” ]
```

```
ALLOWOPENS [ SUBSYS ] process-name
```

```
ALTER EXPORT export-name [ [ , attribute-spec ] ... ]
```

attribute-spec for ALTER EXPORT is:

```
[ ADD-ACCESS [ name                      ]
             [ ( name [ , name ]... )    ] ]
[ DEL-ACCESS [ name                      ]
             [ ( name [ , name ]... )    ] ]
```

```
ALTER GROUP group-name [ [ , attribute-spec ]... ]
```

attribute-spec for ALTER GROUP is:

```
[ ADD-MEMBER { user-name                          }
             { user-name [ , user-name ]... ) } ]
[ DEL-MEMBER { user-name                          }
             { user-name [ , user-name ]... ) } ]
```

```
ALTER LAN lan-name [ [ , attribute-spec ]... ]

    [ , SEL [ NOT ] state ]
```

attribute-spec for ALTER LAN is:

```
[ ADDR-CHECK { ON | OFF } ]
[ ADD-DOMAIN { domain-name                                  }
             { ( domain-name [ , domain-name ]... ) } ]
[ BACKUP backup-cpu-number ]
[ CPU cpu-number ]
[ DEL-DOMAIN { domain-name                                  }
             { ( domain-name [ , domain-name ]... ) } ]
[ HISTOGRAM ( message-length [ , message-length ]...) ]
[ PRI cpu-priority ]
[ PROCESS process-name ]
[ PROGRAM file-name ]
[ SWAP [ file-name ] ]
[ TCPIP-HOST-FILE [ tcpip-host-file ] ]
[ TCPIP-PROCESS-NAME [ tcpip-process-name ] ]
[ TCPIP-RESOLVER-NAME [ tcpip-resolver-name ] ]]
```

```
ALTER NETGROUP netgroup-name [ [ , attribute-spec ]... ]
```

attribute-spec for ALTER NETGROUP is:

```
[ { ADD|DEL } netgroupx-name ]
[ { ADD|DEL } ( [ host-name ] , [ user-name ] ,
           [ domain-name ] ) ]
```

```
ALTER PROCESS process-name [ [ , attribute-spec ]... ]

    [ , SEL [ NOT ] state ]
```

attribute-spec for ALTER PROCESS is:

```
[ BACKUP [ backup-cpu-number ] ]
[ BACKUPDEBUG { ON | OFF } ]
[ COLLECTOR [ process-name ] ]
[ DEBUGONERR { ON | OFF } ]
[ LOGFILE [ file-name ] ]
[ MSGFILE file-name ]
```

```
ALTER SERVER server-name [ [ , attribute-spec ] ... ]
    [ , SEL [ NOT ] state ]
```

attribute-spec for ALTER SERVER is:

```
[ CPU cpu-number ]
[ BACKUP backup-cpu-number ]
[ MAX-FILE-SIZE integer ]
[ MNTPOINT path-name ]
[ NULL-ALIAS-OK { TRUE | FALSE } ]
[ PRI cpu-priority ]
[ PROGRAM file-name ]
[ PROCESS process-name ]
[ READ-ONLY { TRUE | FALSE } ]
[ ROOT-USER-OK { TRUE | FALSE } ]
[ STYPE OSS ]
[ SWAP [ file-name ] ]
[ WRITE-THRU { TRUE | FALSE } ]
```

```
ALTER SUBSYS process-name [ [ , attribute-spec ]... ]

    [ , SEL [ NOT ] state ]
```

attribute-spec for ALTER SUBSYS is:

```
REUSE-SERVERID { ON | OFF }
```

```
ALTER USER  user-name [ [ , attribute-spec ]... ]
```

attribute-spec for ALTER USER is:

```
[ ALIAS OSS { user-name   }
            { user-number } ]
[ COMMENT "string" ]
```

```
DELETE EXPORT export-name
DELETE GROUP group-name
```

```
DELETE LAN lan-name [ , SEL [ NOT ] state ]
```

```
DELETE NETGROUP netgroup-name
```

```
DELETE SERVER server-name  [ , SEL [ NOT ] state ]
```

```
DELETE USER user-name
```

```
INFO EXPORT export-name [ , DETAIL ]
```

```
IINFO GROUP group-name [ , GROUPID group-id ] [ , DETAIL ]
```

```
INFO LAN lan-name [ , DETAIL ] [ , SEL [ NOT ] state ]
```

```
INFO NETGROUP netgroup-name
```

```
INFO PROCESS process-name [ , DETAIL ]

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

     [ , SEL [ NOT ] state ]
```

```
INFO SERVER server-name [ , DETAIL ]

     [ , SEL [ NOT ] state ]
```

```
INFO USER user-name [ , USERID user-id ]

     [ , GROUPID group-id ] [ , DETAIL ]
```

```
LISTOPENS PROCESS process-name
```

```
LISTOPENS SUBSYS process-name [ , DETAIL ]
```

```
NAMES EXPORT export-name
```

```
NAMES GROUP group-name
```

```
NAMES LAN lan-name [ , SEL [ NOT ] state ]
NAMES NETGROUP netgroup-name
```

```
NAMES PROCESS process-name

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

    [ , SEL [ NOT ] state ]
```

```
NAMES SERVER server-name [ , SEL [ NOT ] state ]
```

```
NAMES SUBSYS process-name

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

    [ , SEL [ NOT ] state ]
```

```
NAMES USER user-name
```

```
PRIMARY PROCESS process-name , backup-cpu-number
```

```
START LAN lan-name [ , SEL [ NOT ] state ]
```

```
START SERVER server-name [ , SEL [ NOT ] state ]
```

```
START SUBSYS process-name
    [ , SUB [ ALL | NONE | ONLY | subtype ] ]
    [ , SEL [ NOT ] state ]
```

```
STATS LAN lan-name [ , RESET ] [ , SEL [ NOT ] state ]
```

```
STATS PROCESS process-name [ , RESET ]

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

    [ , SEL [ NOT ] state ]
```

```
STATS SERVER server-name [ , RESET ] [ , SEL [ NOT ] state ]
```

```
STATUS LAN lan-name [ , DETAIL ] [ , SEL [ NOT ] state ]
STATUS PROCESS process-name [ , DETAIL ]

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

     [ , SEL [ NOT ] state ]
```

```
STATUS SERVER server-name [ , DETAIL ]

     [ , SEL [ NOT ] state ]
```

```
STATUS SUBSYS process-name [ , DETAIL ]

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

     [ , SEL [ NOT ] state ]
```

```
STOP LAN lan-name [ , ORDERLY | , FORCED ]

     [ , SEL [ NOT ] state ]
```

```
STOP SERVER server-name [ , ORDERLY | , FORCED ]

     [ , SEL [ NOT ] state ]
```

```
STOP SUBSYS process-name [ , ORDERLY | , FORCED ]

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

     [ , SEL [ NOT ] state ]
```

```
STOPOPENS [ SUBSYS ] process-name
```

```
TRACE LAN lan-name

    { , STOP [ BACKUP ]                                }
    { [ , BACKUP                                  ]
      [ , COUNT count                             ]
      [ , NOCOLL                                  ]
      [ , PAGES pages                             ]
      [ , RECSIZE size                            ]
        , TO file-spec
      [ , WRAP                                    ]
      [ , SELECT { select-word                  } ]
      [ ,        { ( select-word [ , select-word ] ) } ]    }
```

select-word for LAN is:

```
[ ALL ]
[ HOST “host-name” ]
[ MNT { DUMP    }  |  RPC { 100005.2  }
      { EXPORT  }  |      { 100005.5  }
      { MNT     }  |      { 100005.1  }
      { NULL    }  |      { 100005.0  }
      { UMNT    }  |      { 100005.3  }
      { UMNTALL }  |      { 100005.4  } ]
[ NFS { CREATE     }  |  RPC { 100003.9   }
      { GETATTR    }  |      { 100003.1   }
      { LINK       }  |      { 100003.12  }
      { LOOKUP     }  |      { 100003.4   }
      { MKDIR      }  |      { 100003.14  }
      { NULL       }  |      { 100003.0   }
      { READ       }  |      { 100003.6   }
      { READDIR    }  |      { 100003.16  }
      { READLINK   }  |      { 100003.5   }
      { REMOVE     }  |      { 100003.10  }
      { RENAME     }  |      { 100003.11  }
      { RMDIR      }  |      { 100003.15  }
      { ROOT       }  |      { 100003.3   }
      { SETATTR    }  |      { 100003.2   }
      { STATFS     }  |      { 100003.17  }
      { SYMLINK    }  |      { 100003.13  }
      { WRITE      }  |      { 100003.8   }
      { WRITECACHE }  |      { 100003.7   } ]
[ RPC program[(version)].procedure ]
[ RPCMSG ]
[ SERVER server-name ]
[ SPIMSG ]
[ SYSMSG ]
[ UNKNOWN ]
```

```
VERSION LAN lan-name [ , DETAIL ][ , SEL [ NOT ] state ]
```

```
VERSION process-name [ , DETAIL ]

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

      [ , SEL [ NOT ] state ]
```

```
VERSION PROCESS process-name [ , DETAIL ]

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

      [ , SEL [ NOT ] state ]
```

```
VERSION SERVER process-name [ , DETAIL ]

      [ , SEL [ NOT ] state ]
```

B Summary of All SCF Commands

Table B-1 lists the SCF commands and, for each command, indicates whether it is supported by OSS NFS, a general command you can use to manipulate and inquire about SCF, or a general SCF command that is not applicable or useful with OSS NFS.

Table B-1. Summary of SCF Commands (page 1 of 2)

Command	OSS NFS	General SCF	Not Applicable
ABORT	X		
ADD	X		
AGGREGATE			X
ALIAS		X	
ALLOW			X
ALLOWOPENS	X		
ALTER	X		
ASSIGN		X	
ASSUME		X	
BOOT			X
CLEAR		X	
COMMENT		X	
CONFIRM		X	
CONNECT			X
CPUS		X	
DELAY		X	
DELETE	X		
DETAIL		X	
DISCONNECT			X
ENV		X	
EXIT		X	
FC		X	
HELP		X	
HISTORY		X	
INFO	X		
LISTDEV		X	
LISTOPENS	X		

Table B-1. Summary of SCF Commands (page 2 of 2)

Command	OSS NFS	General SCF	Not Applicable
LOAD			X
LOG		X	
MANAGERS		X	
NAMES	X		
OBEY		X	
OPEN		X	
OUT		X	
PAGESIZE		X	
PARAM		X	
PAUSE		X	
PRIMARY	X		
PROBE			X
REPEAT		X	
RUN		X	
SETPROMPT		X	
START	X		
STATS	X		
STATUS	X		
STOP	X		
STOPOPENS	X		
SWITCH		X	
SYSTEM		X	
TIMEOUT		X	
TRACE	X		
VERSION	X		
VOLUME		X	
!		X	
?		X	

Glossary

address mask. A bit mask used to select bits from an Internet address for subnetwork addressing. The mask is 32 bits long. It 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, the resolution might 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 across only a single physical network and is limited to networks that support hardware broadcast.

Advanced Research Projects Agency (ARPA). Now known as the Defense Advanced Research Projects Agency (DARPA).

ARP. *See* Address Resolution Protocol (ARP).

ARPA. *See* Advanced Research Projects Agency (ARPA).

ARPANET. A pioneering long-haul network funded by the Advanced Research Projects Agency (ARPA) (later the Defense Advanced Research Projects Agency or DARPA) and built by Bolt, Beranek, and Newman (BBN). It served as the basis for early networking research as well as for the development of the Internet.

autonomous system. A collection of gateways and networks that fall under one administrative entity and cooperate closely to propagate network reachability (and routing) information among themselves using an interior gateway protocol of their choice. Gateways within an autonomous system have a high degree of trust. At least one gateway in an autonomous system must advertise networks in that system to a core gateway using the Exterior Gateway Protocol (EGP).

baseband. Characteristic of any network technology (like Ethernet) that uses a single carrier frequency and requires all stations attached to the network to participate in every transmission (broadband). *See also* broadband.

best-effort delivery. Characteristic of network technologies that do not provide reliability at link levels. Best-effort delivery systems work well with the Internet, because the Internet protocols assume that the underlying network provides unreliable, connectionless delivery. The combination of Internet Protocol (IP) and User Datagram Protocol (UDP) provides best-effort delivery service to application programs.

big-endian. A format for storage or transmission of binary data in which the most-significant bit or byte is delivered or processed first. The Internet's standard network byte order is big-endian. *See also* little-endian.

bridge. A router that connects two or more networks and forwards packets among them. Usually, bridges operate at the physical network level. Bridges differ from repeaters: bridges store and forward complete packets, while repeaters forward electrical signals.

broadband. A characteristic of any network technology that multiplexes multiple, independent network carriers onto a single cable (usually using frequency-division multiplexing). For example, a single 100-megabits/second (MB/s) broadband cable can be divided into ten 10-MB/second carriers, with each treated as an independent Ethernet. The advantage of broadband is that less cable is needed; the disadvantage is higher cost for equipment.

broadcast. A packet-delivery system that delivers a copy of a given packet to all hosts that attach to it. Broadcast can be implemented with hardware or software.

carrier sense multiple access (CSMA). A characteristic of network hardware that operates by allowing multiple stations to contend for access to a transmission medium by listening to determine whether it is idle.

carrier sense multiple access with collision detection (CSMA/CD). A characteristic of network hardware that uses CSMA combined with a mechanism that allows the hardware to detect when two stations simultaneously attempt transmission. Ethernet is an example of a well-known network based on CSMA/CD technology.

Class A Internet address. An Internet address in which the network number is 1 through 127 (one octet); that is, the first octet of the Internet address, is in the range 1 through 127. The remaining three octets in the address are used for the subnetwork number and host number.

The subnetwork number varies in length. The subnetwork number’s width is typically represented by a bit mask. The rest of the available bits uniquely identify the host connected to the subnetwork. Local area networks (LANs) connected by way of a gateway to the Internet get their subnetwork class from the Network Information Center (NIC). The address classes of stand-alone, or entirely private, LANs are administered by the LAN administrator. The typical usage is to have all Class A Internet addresses for private LANs.

Class B Internet address. An Internet address in which the network number is 128.0 through 191.255 (two octets); that is, the first octet of the Internet address, is in the range 128 through 191, and the second octet is in the range 0 through 255. The remaining two octets in the address are used for the subnetwork number and host number.

The subnetwork number varies in length. The subnetwork number’s width is typically represented by a bit mask. The rest of the available bits uniquely identify the host connected to the subnetwork. Local area networks (LANs) connected by way of a gateway to the Internet get their subnetwork class from the Network Information Center (NIC). The address classes of stand-alone, or entirely private, LANs are administered by the LAN administrator. The typical usage is to have all Class A Internet addresses for private LANs.

Class C Internet address. An Internet address in which the network number is 192.0.0 through 255.255.255 (three octets); that is, the first octet of the Internet address, is in the range 192 through 255, the second octet is in the range 0 through 255, and the third octet is in the range 0 through 255. The remaining octet in the address is used for the subnetwork number and host number.

The subnetwork number varies in length. The subnetwork number's width is typically represented by a bit mask. The rest of the available bits uniquely identify the host connected to the subnetwork. Local area networks (LANs) connected by way of a gateway to the Internet get their subnetwork class from the Network Information Center (NIC). The address classes of stand-alone, or entirely private, LANs are administered by the LAN administrator. The typical usage is to have all Class A Internet addresses for private LANs.

Compaq NonStop Kernel Open System Services (OSS). The product name for the OSS environments. *See also* Open System Services (OSS).

Compaq NonStop TCP/IP. Transmission Control Protocol over Internet Protocol that runs on a Compaq NonStop Himalaya system. Conventional NonStop TCP/IP, formerly named Tandem TCP/IP, uses the conventional NonStop TCP/IP stack. Parallel Library TCP/IP uses parallel TCP/IP protocol stacks implemented as a Shared Runtime Library (SRL) on the NonStop Kernel operating system.

connection. The path between two protocol modules that provides reliable stream delivery service. In the Internet, a connection extends from a Transmission Control Protocol (TCP) module on one system to a TCP module on another system.

connectionless service. A 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.

core gateway. One of a set of gateways operated by the Internet Network Operations Center (INOC) at Bolt, Baranek, and Newman (BBN). Gateways in the core system exchange routing updates periodically to ensure that their routing tables remain constant. The core forms a central part of Internet routing, because all groups must advertise paths to their networks to core gateways using the Exterior Gateway Protocol (EGP).

CSMA. *See* carrier sense multiple access (CSMA).

CSMA/CD. *See* carrier sense multiple access with collision detection (CSMA/CD).

DARPA. Defense Advanced Research Projects Agency. Formerly Advanced Research Projects Agency (ARPA).

directory. In an NFS fileset, a file that contains a list of other files, including other directories that are below it in the hierarchy. Entries in a directory file are called links. Each directory contains at least two links, `.` (dot) and `..` (dot-dot). The link called "dot" points to the directory itself, and the link called "dot-dot" points to the parent.

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).

dotted-decimal notation. A representation for a 32-bit binary integer that consists of four decimal (base 10) numbers separated by periods (dots), each of which represents one of the four 8-bit values in the integer. Many Internet application programs accept dotted-decimal notation in place of destination machine names.

ECHO. The name of a program used in the Internet to test the reachability of destinations by sending them an Internet Control Message Protocol (ICMP) echo request and waiting for a reply.

Ethernet. A popular local area network (LAN) technology invented at the Xerox Corporation Palo Alto Research Center. An Ethernet connection itself is a passive coaxial cable; the interconnections contain all active components. Ethernet is a best-effort delivery system that uses carrier sense multiple access with collision detection (CSMA/CD) technology. Xerox Corporation, Digital Equipment Corporation, and Intel Corporation developed and published the standard for a 10-megabits/second Ethernet.

Ethernet meltdown. An event that causes saturation or near saturation on an Ethernet. It usually results from illegal or misrouted packets and typically lasts only a short time. As an example, consider an Internet Protocol (IP) datagram directed to a nonexistent host and delivered by way of hardware broadcast to all machines on the network. Gateways receiving the broadcast will send out Address Resolution Protocol (ARP) packets in an attempt to find the host and deliver the datagram.

external data representation (XDR). The standard for a machine-independent data structure representation developed by Sun Microsystems, Inc. To use XDR, a sender translates from the local machine representation to the standard external representation and a receiver translates from the standard external representation to the local machine representation.

FDDI. *See* fiber distributed data interface (FDDI).

fiber distributed data interface (FDDI). An emerging standard for a network technology based on fiber optics. FDDI specifies a 100-megabits/second data rate using 1300-nanometer light wavelength and limits networks to approximately 200 kilometers (km) in length, with repeaters every 2 km or less. The access control mechanism uses token-ring technology.

filename. In the OSS environment, a component of a pathname containing any valid characters other than a slash (`/`) or null. In the Guardian environment, a filename is the set of node name, volume name, subvolume name, and file identifier characters that uniquely identifies a file.

file server. A process running on a computer that allows programs running on remote machines to access files on that computer. The term is often applied loosely to computers that run file-server programs.

fileset. A logical grouping of files that, except for the root of the fileset, can be contained only by directories within the fileset. This manual uses the term “fileset” for consistency with other Open System Services documentation. *See also* file system.

file system. In the OSS environment, a collection of files and file attributes. A file system provides the namespace for the file serial numbers that uniquely identify its files. Open System Services provides a file system (see also ISO/IEC IS 9945-1:1990 [ANSI/IEEE Std. 1003.1-1990], Clause 2.2.2.38); the Guardian environment provides a file system; and OSS NFS provides a file system. (OSS NFS filenames and pathnames are governed by slightly different rules than OSS filenames and pathnames.) Within the OSS and OSS NFS file systems, filesets exist as manageable objects.

On a Himalaya system, the Guardian file system for a node is a subset of the OSS virtual file system. Traditionally, the application program interface (API) for file access in the Guardian environment is referred to as the "Guardian file system."

In some UNIX and NFS implementations, the term "file system" is used to mean the same thing as "fileset." That is, a file system is a logical grouping of files that, except for the root of the file system, can be contained only by directories within the file system. *See also* fileset.

file transfer protocol (FTP). (1) The Internet-standard, high-level protocol for transferring files from one system to another. The server side requires the client to supply a logon identifier and password before it honors requests. FTP makes no assumptions about the file-naming structure of the source and destination systems, and it allows the file names of each system to be represented in the vernacular.

(2) The application used to send complete files over Transmission Control Protocol/Internet Protocol (TCP/IP) services.

FTP. See file transfer protocol (FTP).

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 Internet Protocol (IP) datagrams among the networks it is connected to. Gateways route packets to other gateways until they can be delivered to the final destination.

Gateway-to-Gateway Protocol (GGP). The protocol that core gateways use to exchange routing information. GGP implements a distributed shortest-path routing algorithm. Under normal circumstances, all GGP participants reach a steady state in which the routing information at all gateways agrees.

GGP. *See* Gateway-to-Gateway Protocol (GGP).

Guardian. An environment available for interactive or programmatic use with the Compaq NonStop Kernel operating system. Processes that run in the Guardian environment use the Guardian system procedure calls as their application program interface; interactive users of the Guardian environment use the Compaq Tandem Advanced Command Language (TACL) or another Compaq product's command interpreter. *Contrast with* Open System Services (OSS).

hard link. The relationship between two directory entries for the same file. A hard link acts as an additional pointer to a file. A hard link cannot be used to point to a file in another fileset. *Compare to* symbolic link.

hardware address. *See* media access control (MAC) address.

hierarchical routing. Routing based on a hierarchical addressing scheme. Most Internet routing is based on a two-level hierarchy in which an Internet address is divided into a network portion and a host portion. Gateways use only the network portion until the datagram reaches a gateway that can deliver it directly. Subnetworking introduces additional levels of hierarchical routing.

hop count. A measure of distance between two points in the Internet. A hop count of *n* means that *n* gateways separate the source and destination.

ICMP. *See* Internet Control Message Protocol (ICMP).

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, Routing Information Protocol (RIP) is among the most popular.

Internet. The collection of networks and gateways, including the ARPANET, MILNET, and NSF network, that use the Transmission Control Protocol/Internet Protocol (TCP/IP) suite and function as a single, cooperative virtual network. The Internet provides universal connectivity and three levels of network service: unreliable, connectionless packet delivery; reliable, full-duplex stream delivery; and application-level services like electronic mail that build on the first two.

Internet address. The 32-bit address assigned to hosts that want to participate in the Internet using Transmission Control Protocol/Internet Protocol (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. This 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. This protocol is used by the network layer to communicate the reachability of particular network nodes as well as routing control information. ICMP is part of IP because it shares the same Ethernet type field.

Internet Protocol (IP). 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. (1) Within a Himalaya node, the ability to use the features or facilities of one environment from another. For example, the `gtacl` command in the OSS environment allows an interactive user to start and use a Guardian tool in the Guardian environment.

(2) Among systems from multiple vendors or with multiple versions of operating systems from the same vendor, the ability to exchange status, files, and other

information. NonStop Himalaya manuals often use the term “connectivity” in this context, while other vendors use the term “connectivity” to mean hardware compatibility.

IP. *See* Internet Protocol (IP).

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. International Organization for Standardization. ISO is an international body that drafts, discusses, proposes, and specifies standards for network protocols. ISO is best known for its seven-layer reference model that describes the conceptual organization of protocols.

ISO is sometimes called the “International Standards Organization”; although ISO is the official abbreviation, it does not correspond to the organization’s name in any language.

LAN. *See* local area network (LAN).

level 2. A reference to LINK LEVEL communication (for example, frame formats) or link-level connections derived from the ISO seven-layer reference model. For long-haul networks, level 2 refers to the communication between a host computer and a network packet switch (for example, use of high-level data link control/link access procedure balanced or HDLC/LAPB). For local area networks (LANs), 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 seven-layer reference model. For the Internet, level 3 refers to the Internet Protocol (IP) and IP datagram formats. Thus, a level 3 address is an Internet address.

link name. The filename associated with a specific file within a directory. The length of a filename, and therefore the length of a link name, depends on the file system. Within the NFS file system, link names can be up to 255 characters long and can contain any ASCII characters except for the NUL (binary 0) and the slash (`/`); such link names are case-sensitive. *See also* filename.

little-endian. A format for storage or transmission of binary data in which the least-significant bit or byte is delivered or processed first. *See also* big-endian.

local area network (LAN). Any physical network technology that operates at high speed (usually tens of megabits/second to several gigabits/second) over short distances (up to a few thousand meters). Examples include Ethernet and proNET-10.

local mount. A mount that attaches the fileset associated with a server to the specified mount point within the local directory hierarchy. The local mount is visible within the OSS NFS subsystem and makes the files associated with the server available through the path associated with the local mount point.

logical name. In NFS, a case-insensitive alphanumeric name that starts with a letter and contains eight or fewer characters. Logical names are used to name LAN and SERVER objects.

MAC address. *See* media access control (MAC) address.

media access control (MAC) address. A 12-digit hexadecimal number used as an address for network access. Each Compaq Tandem LAN Access Method (TLAM) controller is assigned a unique MAC address, which is burned into its programmable read-only memory (PROM) as part of the manufacturing process. Compaq, like other manufacturers of IEEE 802.3 controllers, is licensed to use a dedicated range of MAC addresses. That way, each controller is guaranteed its own unique address.

mount point. A directory that contains a mounted fileset. The mounted fileset can be in a different file system.

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 to receive a copy of a packet consists of the entire set.

Network File System (NFS). A protocol developed by Sun Microsystems, Inc. that uses Internet Protocol (IP) to allow a set of cooperating computers to access each other's filesets as if they were all 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 namespace. NFS is used primarily on UNIX systems, but it has been implemented for many systems, including personal computers like the IBM PC and Apple Macintosh computer.

NFS. *See* Network File System (NFS).

NonStop TCP/IP. *See* The product name for the OSS environments. See also Open System Services (OSS)..

object. A programming procedure that provides named NFS services. NFS objects are created by Subsystem Control Facility (SCF) commands.

Open System Services (OSS). A programming and software-execution environment that provides UNIX-like facilities on a Himalaya system and is available for interactive or programmatic use with the NonStop Kernel operating system. Processes that run in the OSS environment use the OSS application program interface; interactive users of the OSS environment use the OSS shell for their command interpreter. *Contrast with* Guardian.

Open Systems Interconnection (OSI). A reference to protocols, specifically ISO standards, for the interconnection of cooperative computer systems.

OSI. *See* Open Systems Interconnection (OSI).

OSS. *See* Open System Services (OSS)

Parallel Library TCP/IP. A Compaq networking product that provides increased performance, scalability, and fault-tolerance by creating parallel TCP/IP protocol stacks implemented as an Shared Runtime Library (SRL) on the NonStop Kernel operating system.

pathname. A name that identifies a file by specifying which directories must be searched in order to find the link to the file. A pathname consists of zero or more directory names, separated by slash (/) characters, and followed by a filename. Multiple consecutive slashes in a pathname are treated as a single slash. If the pathname begins with a slash, the search begins with the root directory; otherwise, the search begins with the current directory. There is no concept of current directory with NFS Subsystem Programmatic Interface, or SPI, commands, so all pathnames passed as SPI tokens must begin with a slash. The maximum pathname length is 1024 characters.

PCNFSD. *See* (PC)NFS Daemon (PCNFSD).

(PC)NFS Daemon (PCNFSD). The name of a program that provides user authentication and NFS printing services to PCs and other workstations unable to provide UNIX authentication information.

PING. Packet InterNet Groper. *See* ECHO.

remote mount. A mount used by an NFS client to attach part of the local NFS file hierarchy to a point within the client's remote file hierarchy. The remote mount is visible only to the NFS client performing the mount. In effect, the local hierarchy from the mount point down is exported to the client performing the remote mount.

remote procedure. A procedure or function packaged to be called within a server process indirectly by a client process.

Remote Procedure Call (RPC). The action of calling a remote procedure. RPC provides a standard way to invoke services operated on other servers that are linked by a network (remote). The Remote Procedure Call (RPC) specification is defined by RFC (Request for Comment) 1057.

Request for Comments (RFC). The 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.

RFC. *See* Request for Comments (RFC).

RPC. *See* Remote Procedure Call (RPC).

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 that they exchange to transfer mail.

SMTP. *See* Simple Mail Transfer Protocol (SMTP).

SNAP. *See* Subnetwork Access Protocol (SNAP).

socket. An end-point for stream-oriented communication. A socket has a file descriptor.

source quench. A congestion-control technique in which a system experiencing congestion sends a request back to the source of the packets causing the congestion asking that the source stop transmitting. In the Internet, gateways use Internet Control Message Protocol (ICMP) source quench to stop or reduce the transmission of Internet Protocol (IP) datagrams.

Subnetwork Access Protocol (SNAP). Defines the interface between the Internet Protocol (IP) layer and the link-level control (LLC) layer to run the Transmission Control Protocol/Internet Protocol (TCP/IP) suite over IEEE networks. The interface is accomplished by 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 destination SAP (DSAP) fields of the LLC header.

subnetwork work 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 subnetwork work addressing, routing continues as usual by dividing the destination address into an Internet portion and local portion. Gateways and hosts inside a site using subnetwork work addressing interpret the local portion of the address by dividing it into a physical network portion and host portion.

symbolic link. A type of special file that acts as a name pointer to another file. A symbolic link contains a pathname and can be used to point to a file in another fileset. Symbolic links are not included in ISO/IEC IS 9945-1: 1990. *Compare to* hard link.

Tandem TCP/IP. *See* The product name for the OSS environments. See also Open System Services (OSS)..

TCP. *See* Transmission Control Protocol (TCP).

TCP/IP. *See* Transmission Control Protocol (TCP) over Internet Protocol (IP).

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 though the user’s terminal were connected directly to the remote system. That is, the user invokes a TELNET application program that connects to a remote system, prompts for a login ID and password, and then passes keystrokes from the user’s terminal to the remote system and displays output from the remote system on the user’s terminal. TELNET provides a virtual terminal service. Through the concept of a Network Virtual Terminal (NVT), it allows clients to access remote applications through use of a remote server. TELNET has many options and is quite flexible. Through it, many diverse terminal data streams can be communicated beyond the basic NVT data stream.

Transmission Control Protocol (TCP). The Internet-standard transport-level protocol that provides the reliable, full-duplex stream service that many application protocols depend on. TCP allows a process on one system to send a stream of data to a process on another. It is connection-oriented; that is, before transmitting data, participants must establish a connection.

Software implementing TCP usually resides on the operating system and uses the Internet Protocol (IP) 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. The TCP protocol is used by applications that require reliable end-to-end data transfer. The TCP protocol is a byte-stream-oriented protocol that includes no concept of packet boundaries; the only guarantee is that all of the data sent will be received in the same order in which it was sent.

Transmission Control Protocol over Internet Protocol (TCP/IP). *See* Transmission Control Protocol (TCP) *and* Internet Protocol (IP).

UDP. *See* User Datagram Protocol (UDP).

User Datagram Protocol (UDP). The Internet-standard protocol that allows an application program on one system to send a datagram to an application program on another system. UDP uses the Internet Protocol (IP) 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 system. In practice, UDP also includes a checksum over the data being sent.

UDP provides unreliable datagram service. The integrity of the packets of data sent is maintained; however, the delivery of the packet is not guaranteed. When a packet is received, though, it is guaranteed to be exactly what was sent from the remote site. Also, the ordering of datagrams is not guaranteed; it is possible to receive packets out of order when using UDP.

XDR. *See* external data representation (XDR).

Index